AFRL-IF-RS-TR-2002-5
Final Technical Report
January 2002

ADAPTIVE SYSTEM SECURITY POLICIES

The Boeing Company

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J636

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

20020405 035

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-5 has been reviewed and is approved for publication.

Glen E. Bahr

APPROVED: GLEN E. BAHR
Project Engineer

FOR THE DIRECTOR: WARREN H. DEBANY, Jr.
Technical Advisor
Information Grid Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFGB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** Jan 02	**3. REPORT TYPE AND DATES COVERED** Final Jul 97 - Apr 00

4. TITLE AND SUBTITLE ADAPTIVE SYSTEM SECURITY POLICIES	**5. FUNDING NUMBERS** C - F30602-97-C-0217 PE - 63760E PR - F256 TA - 40 WU - 02
6. AUTHOR(S) Dan Sachnackenberg, Kelly Bunn, Daylan Darby, Laurence Rockwell, Travis Reid, Dan Sterne, Kelly Djahandari, Brett Wilson, Karl Levitt and Jeff Rowe	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The Boeing Company Phantom Works PO Box 3999 Seattle, WA 98124-2499	**8. PERFORMING ORGANIZATION REPORT NUMBER** D950-10468-1
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFGB 525 Brooks Road Rome, NY 13441-4505	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2002-5

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Glen E. Bahr, IFGB, 315-330-3515

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

This contract final technical report documents the Adaptive System Security Policies project results. This project extended concepts developed in the Dynamic Cooperating Boundary Controllers project, which developed the initial version of the Intruder Detection and Isolation Protocol (IDIP). IDIP provides an infrastructure for intruder tracking and containment. The focus of the extensions developed under the Adaptive System Security Policies was to provide some policy control over IDIP responses to ensure that the damage caused by the response is no more than the damage caused by the intruder. This report provides an overview of the current IDIP implementation and focuses on policy control for automated response.

14. SUBJECT TERMS Dynamic Cooperating Boundary Controllers, Intruder Detection and Isolation Protocol, IDIP, automated response, intrusion, computer security	**15. NUMBER OF PAGES** 96
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

CONTENTS

Page
1. INTRODUCTION 1
1.1 Background 1
1.2 Approach 2
1.3 Summary of Accomplishments 5
1.4 Scope 9
2. IDIP DESIGN AND IMPLEMENTATION 10
2.1 IDIP Overview 10
2.1.1 IDIP Application-Layer Protocol 16
2.1.2 IDIP Message Layer 18
2.1.2.1 Neighborhood Management 18
2.1.2.2 Key Distribution 18
2.1.2.3 Cryptographic Extensions to the Message Layer 18
2.2 IDIP Software Architecture 20
2.2.1 IDIP Backplane 20
2.2.2 IDIP Applications 22
2.2.2.1 IDIP Generic Agent 22
2.2.2.2 Custom IDIP Responders 24
2.2.2.3 Discovery Coordinator Applications 25
2.3 Representing Intrusion Detection and Response Information 29
2.4 Policy Mechanisms 33
2.4.1 Requirements 33
2.4.2 IDIP Nodes and Roles 33
2.4.3 Parameters Used for Response Policy 34
2.4.4 Concepts 35
2.4.5 Cost Models 36
2.4.5.1 Generic IDIP Agent Cost Model Implementation 36
2.4.5.1.1 Detector Agent Model 36
2.4.5.1.2 Response Agent Model 38
2.4.5.2 Cost Model Implementation in the Discovery Coordinator 38
2.4.6 Architecture for Detection and Response Policy Projection 40
2.4.6.1 NAVM 41
2.4.6.2 Computing the NAVM and Projecting Detection and Response Policy 43
2.4.6.3 Policy projection for Rule Based IDIP agents 46
2.4.6.4 Dynamic Policy Projection 46
2.4.6.5 Dynamic Models 46
2.4.7 Multi-Community Policy Issues 47
2.4.7.1 Inter-Community Cooperation 47
2.4.7.1.1 Outgoing Message Sanitization 48
2.4.7.1.2 Incoming IDIP Message Modification 49
2.4.7.1.3 Inter-Community Differences in Protection Policies 50

CONTENTS (CONTINUED)

Page

2.4.7.1.4 Inter-Community Differences in Attack Definition 50
2.4.7.2 Inter-Community Certification Authority (CA) 50
2.5 Changes to IDIP 51
2.6 Analysis 52
3. PROJECT ACCOMPLISHMENTS 54
3.1 Overall Accomplishments 54
3.2 Capabilities Developed 54
4. SYSTEM-LEVEL DEMONSTRATION 56
4.1 Experiment Configuration and Scenario 56
4.2 Initial IFE 2.3 58
4.2.1 Integration 58
4.2.1.1 User-Based Responses 58
4.2.1.2 Component-Specific Responses 59
4.2.1.3 MPOG Responses 59
4.2.1.4 IP Filter and TCP Wrappers 59
4.2.1.5 Discovery Coordinator Cost Model 60
4.2.1.6 Cyber Command System Integration 60
4.2.2 Experiment Execution 60
4.2.2.1 Red Team Attack 60
4.2.2.2 Detection and Response 63
4.2.2.3 Issues 63
4.3 IFE 2.3 Rerun 64
4.3.1 Integration 64
4.3.2 Experiment Execution 65
4.3.2.1 Red Team Attacks 66
4.3.2.2 Detection and Response 66
4.4 Conclusions 67
5. CIDF DEMONSTRATION 68
6. SUMMARY AND CONCLUSION 70
6.1 Lessons-Learned 70
6.2 Further Investigations, Research, and Development 71
6.3 Conclusion 72
7. REFERENCES 75
A. APPENDIX A MESSAGE LAYER API SPECIFICATION 77
A.1 Definitions 77
A.2 Start-Up Calls 78
A.2.1 CIDFMLinit 78
A.2.2 CIDFMLbind 78
A.2.3 CIDFMLrestrict 79

CONTENTS (CONTINUED)

Page

A.3 Communication Calls 79
A.3.1 CIDFMLrecvfrom 79
A.3.2 CIDFMLsendto 80
A.4 Termination Calls 81
A.4.1 CIDFMLclose 81
A.4.2 CIDFMLexit 81
A.5 Error Codes 82

LIST OF FIGURES

Page
Figure 1-1. IDIP Nodes 2
Figure 2-1. IDIP Communities 11
Figure 2-2. Attack Scenario 12
Figure 2-3. IDIP Local Neighborhood Response 13
Figure 2-4. IDIP Remote Boundary Controller Response 13
Figure 2-5. IDIP Remote Boundary Controller Response (Continued) 14
Figure 2-6. IDIP Intrusion Reporting 14
Figure 2-7. IDIP Discovery Coordinator Optimal Response 15
Figure 2-8. IDIP Backplane Architecture 21
Figure 2-9. IDIP Generic Agent Architecture 22
Figure 2-10. Discovery Coordinator Application View 26
Figure 2-11. Topology Format 27
Figure 2-12. Example Network Configuration 28
Figure 2-13. Example Topology Specification 29
Figure 2-14. CISL Example 30
Figure 2-15. Automatically Propagating GrIDS-Style Worm 31
Figure 2-16. CISL-Style Worm Description 32
Figure 2-17. Detection and Response Policy Components 41
Figure 2-18. Computation of NAVM 44
Figure 2-19. Default Requirement Strength Definitions 45
Figure 2-20. Community Relationships 48
Figure 4-1. IFE 2.3 Configuration 57
Figure 4-2. IFE 2.3 Attack Tree 61
Figure 4-3. IFE 2.3 Attack Tree 61
Figure 4-4. IFE 2.3 Attack Tree 63
Figure 4-5. IFE 2.3 Attack Tree 63
Figure 5-1. CIDF Demonstration Configuration 69
Figure 6-1. Current Cryptographic Algorithms 74

GLOSSARY

API	application programmer's interface
AH	authentication header
ACK	acknowledgment
BSM	Basic Security Module
CA	certification authority
CBC	cipher block chaining
CIDF	Common Intrusion Detection Framework
CISL	Common Intrusion Specification Language
CORBA	Common Object Request Broker Architecture
COTS	commercial off the shelf
DC	Discovery Coordinator
DEFPOS	cyber defense posture
EMERALD	Event Monitoring Enabling Responses to Anomalous Live Disturbances
ESP	encapsulating security protocol
FTP	File Transfer Protocol
HMAC	hashed message authentication code
HTTP	Hyper-Text Transfer Protocol
IDIP	Intruder Detection and Isolation Protocol
IDS	intrusion detection system
IFE	integrated feasibility experiment
INFOCON	Information Condition
IP	Internet Protocol
IPSec	IP Security
LAN	local area network
MPOG	Multi-Protocol Object Gateway
NAVM	network asset value model
NKID	neighborhood key information distribution
SHA	Secure Hash Algorithm
SID	semantic identifier
SMARTS	System Management and Administration for Remote Trusted System
SNMP	Simple Network Management Protocol
SSL	Secure Socket Layer
SYN	TCP's synchronization flag
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	wide area network

Adaptive System Security Policies

1. INTRODUCTION

The Intruder Detection and Isolation Protocol (IDIP) was developed to support real-time tracking and containment of attacks that cross network boundaries. This report describes proposed extensions to IDIP and the IDIP concept of operations that enable more effective intrusion response. The IDIP concept of operations has each response component independently deciding on what is an appropriate response. The system's objective is to generate the response as close as possible to the attacker, minimizing the response impact on the critical functions of the system under attack. Each component's objective is to allow this optimal response while protecting local resources as well. This report discusses improvements to how IDIP nodes respond through better cooperation and a more consistent set of response mechanisms.

The primary notion that has evolved through this effort is that IDIP responses should be taken in two stages: (1) an initial immediate response that may be relatively harsh (i.e., may cause damage to normal system functionality), but is relatively short-lived, and (2) a more reasoned "optimal" response that is more effective at meeting the system's overall operational needs while attempting to contain the attack.

1.1 Background

In the Dynamic, Cooperating Boundary Controllers contract, we developed IDIP and the basic IDIP concept of operations that enables intruder tracking and containment. However, the response mechanisms developed were generally harsh (e.g., blocking incoming TELNET connections until manual intervention), potentially causing more damage to the system than the original attack. Through that effort, we developed an infrastructure that can support better response mechanisms, and with some extensions to the protocol and response components, can provide optimal responses that can adapt to changes in the network threat environment.

Figure 1-1 shows the various components that can participate in an IDIP-based response. Intrusion detection components initiate IDIP response messages, and can support damage assessment and recovery within the local environment. Boundary controllers (e.g., routers and firewalls) provide network-based response mechanisms by blocking the intruder's access to network resources. Hosts provide finer-grained responses by killing processes and connections associated with intruders. A centralized network management component (called the Discovery Coordinator) receives intrusion reports and audit data from other IDIP nodes, enabling it to (1) provide administrative personnel with a global picture of the system intrusion status and (2) coordinate the overall system response to attacks.

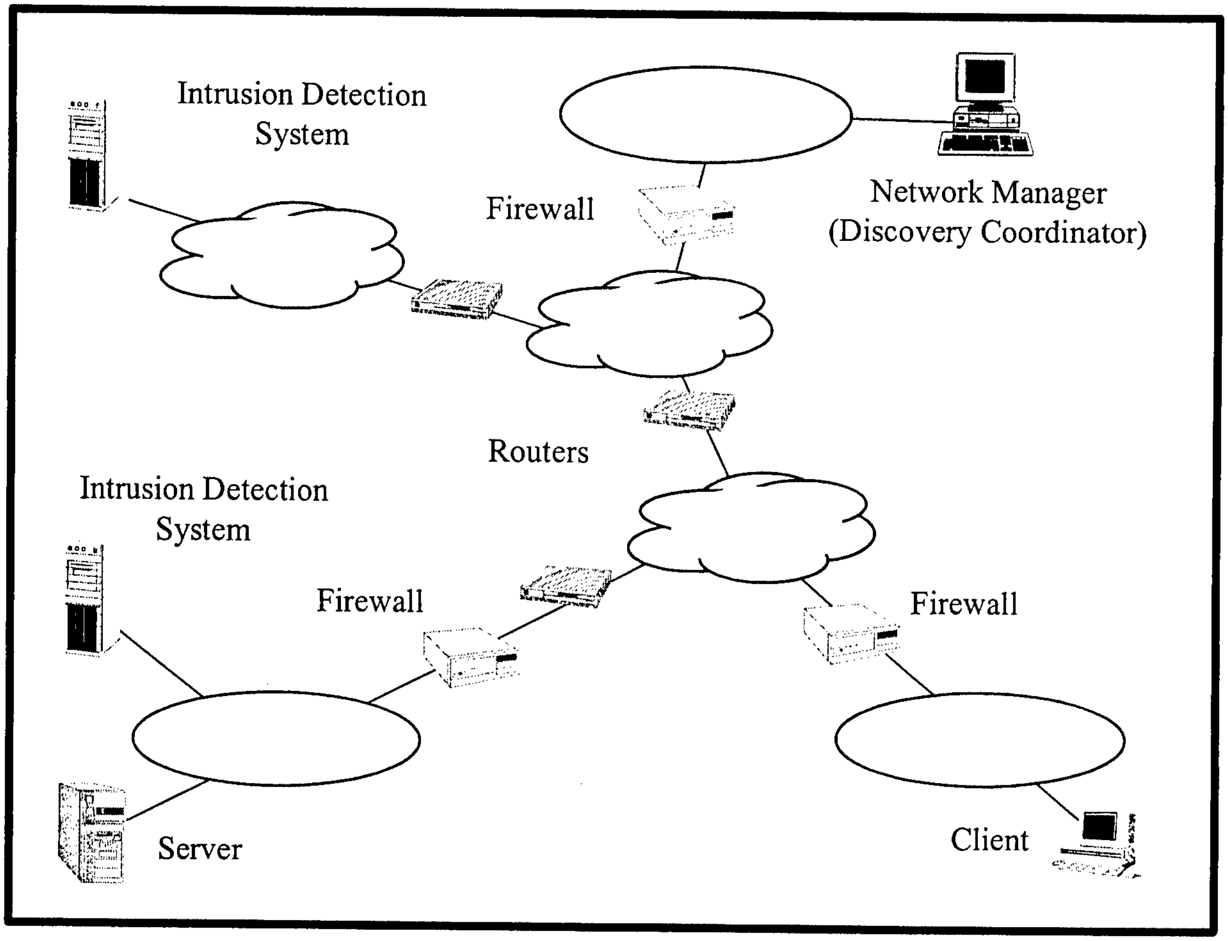

Intrusion Detection System
Network Manager (Discovery Coordinator)
Firewall
Routers
Intrusion Detection System
Firewall
Firewall
Server
Client

Figure 1-1. IDIP Nodes

1.2 Approach

We used the same underlying requirements developed during the Dynamic, Cooperating Boundary Controllers project, including providing support for responses across heterogeneous networks of networks that may span administrative boundaries. This implies that the mechanisms developed cannot be tightly integrated between different components. For example, no component knows enough about the system to determine the precise responses required. Because of the synergy gained from the Information Assurance Automatic Response to Intrusion contract, we were able to develop and demonstrate more capability than originally planned.

Our approach was to investigate extensions to IDIP in four general areas: (1) generation of locally optimal responses based on the system policies that define what is critical for the system to survive; (2) adaptation of the policies as the system threat level changes; (3) development of more intelligent control mechanisms at the Discovery Coordinator to enable generation of globally optimal responses; and (4) development of mechanisms to support response across different (minimally cooperative) administrative domains.

Our investigation includes attempting to categorize the most appropriate responses for various classes of attack.

We categorized types of responses as follows-

a. None.

b. Prevent (filtering rules).

c. Tracing.

d. Notify situation assessment mechanisms.

e. Increase monitoring.

f. Stop attacker current attack (e.g., kill connection or process).

g. Stop attacker future attacks (e.g., add filtering rules to boundary controllers, disable user account, or halt the IDIP node).

h. Padded cell. (i.e., alter the system behavior as perceived by attacker. For example, a component might intercept the response to a failed attempt to penetrate the finger daemon, and alter the perceived response so that attack looks like it has succeeded when it actually has been foiled. It might also be useful to deflect (misroute) the attack to another network or set of hosts to keep the attacker engaged.)

i. Limit attack (e.g., slow down a process or network traffic).

j. Disrupt attack. (In a staged attack, stopping one of the stages can be sufficient to stop the attack. For example, in Transmission Control Protocol (TCP) sequence number guessing, if the response system generates additional connections with the target at the right time, the guessed sequence number will be incorrect.)

k. Recovery. (An example of a real-time recovery mechanism is Purdue's SynKill Server, which sends a SYN ACK to server so that server doesn't wait for a SYN ACK from the attacker. The attacker must then exhaust (virtual) memory of server machine, rather than exhaust half-completed connection table. The SynKill Server also remembers source addresses used in SYN attacks, and for some time period, prevents those addresses from being used by sending a TCP reset when a connection is attempted.)

l. Attack back. Attacking the machine that is the source of an attempted intrusion in general falls outside the scope of this project. However, when the attack source is within the IDIP system's administrative domain, attacking back becomes a reasonable defensive action when no other mechanism exists to stop the attack.

We also categorized attacks by the type of responses that appear most appropriate. In all cases, the response should include tracing back to the attack source, and if possible, reaching into the host that is the attack source and killing the attacking process. We did not consider the use of padded cells in this analysis because they are very difficult to develop as a generic mechanism, and so are not readily available as a response mechanism during the course of this contract.

a. Flooding. For flooding attacks, the appropriate response is to block the source of the flood unless the flood is using a critical service. In that case, slowing down the source is more appropriate, however this also causes the system to slow down potentially useful traffic. This is still better than blocking the traffic. Because flooding can use any address and port combination, it is difficult (if not impossible) to distinguish useful communication from forged flood packets. Short of disabling an interface, it is difficult to provide other responses unless IDIP traces all the way back to the malicious process and kills the process. Note that in slowing down the attacks on the network, response components must discard some of the packets, including potentially useful ones because network components cannot use buffering to cause the delay. Using buffering requires the response component to buffer all incoming traffic and forward it at a rate lower than the incoming rate. This causes internal buffers to quickly fill requiring dropped packets.

b. Staged attacks. For staged attacks, the least harmful response is to disrupt the attack through mechanisms such as additional packet insertion or very short-term service blocking. For example in many attacks, one typically probes to gather information prior to attempting the specific attack. If the probing is detected, the response system can briefly block an interface or insert packets to confuse the attacker. Increased monitoring may be required to see the full attack scenario.

c. Invalid operations, including denial of service attacks that crash system components. For attacks such as the "Ping of Death" where a single packet can crash a target system, the best approach is prevention. Typically, many of these attacks are easily recognized and can be blocked at filtering routers or firewalls. However, for various reasons (including performance penalties of filtering), many boundary controllers are not configured to block these attacks. The next best response is to initiate attack blocking for some reasonably long time period until the attack subsides.

d. Other attacks. Block the datagrams involved in the attack as narrowly as possible so that as much legitimate traffic as possible can pass through the network. This blocking should be short-lived while the system defines a more optimal response.

Providing responses in hosts provides some added flexibility. If the host can reliably respond, it can-

a. Kill the connection.

b. Kill the process.

c. Disable suspected user accounts.

d. Slow down the process.

e. Reduce user or process privileges.

Response components at the edge of the IDIP network can also attempt to acquire additional information about the attack source and forward that information to the Discovery Coordinator. This information could include reverse finger, trace route, or query to the authentication server.

In each case, the information may be incorrect depending on the extent to which the attacker has compromised the infrastructure, but it can provide an administrator with more information on the attack.

1.3 Summary of Accomplishments

We have completed development of adaptive policy mechanisms for each detection and response agent, and the Discovery Coordinator. These have been tested in Boeing's Security Technology lab; however, we were unable to demonstrate their use in the integrated feasibility experiment (IFE) 2.3 (which was to serve as a final demonstration of the IDIP adaptive policy mechanisms). During the IFE, most of the IDIP agent response mechanisms were used, however, due to a setup error, the policy mechanisms at the Discovery Coordinator were not used. Various aspects of the policy mechanisms were also demonstrated at IFE 2.1 against an adversary modeled after the hacker community. The mechanisms behaved as expected in the experiment; however, the topology was not sufficiently complex to provide any insight into how well the Discovery Coordinator mechanisms work. Additionally, since that time, a few modifications and new features were added to the policy mechanisms.

IFE 2.3 was a red-team experiment; however, the environment was constructed in a manner that allowed the red team to accomplish their objectives without triggering any detectors. Because there were no detected events, the response policy mechanisms were not used. Subsequently, the environment has been modified to improve the likelihood of detection through adding new custom detectors. The IFE was re-run in December 1999 and at that time, demonstrated that the IDIP agent policy mechanisms are effective in altering adversary behavior, increasing their exposure, and increasing their work factor.

The IDIP policy prototype implementation uses a concept of a "cost model" that assigns value to network services and uses those values to determine whether a response is warranted. There are three separate cost models: (1) detector; (2) responder; and (3) Discovery Coordinator. The detector and responder cost models have only a local view of the situation and resource values, while the Discovery Coordinator has a global view. These models and their implementation in the IDIP system address policy issues as follows.

a. **Performance impact.** The cost models are only executed during system attack, so they have no impact on normal operational performance. They are also relatively efficient mechanisms for determining response at both the detector and response agents. In these components, the mechanism requires matching attack information against a table of costs in a manner similar to what is done for applying filtering rules in a firewall or router. This can be made to operate very efficiently. At the Discovery Coordinator, the performance is slower, as the Discovery Coordinator searches for an optimal response. In simple test cases, however this was also relatively efficient. The Discovery Coordinator performance can also be mitigated through the duration values used in response components. The default blocking rules are for 2 minutes - enough time for the Discovery Coordinator with the cost models developed thus far. Longer default blocking times could be used for slower algorithms (or larger models).

b. **Topology limitations.** The local IDIP agents are limited by their local topology knowledge and the attack path. So local IDIP nodes can only block an attack along the attack path. However, the Discovery Coordinator has access to the full network topology. The Discovery Coordinator provides a good location for determining optimal responses as it has knowledge about alternate network paths that must be blocked to contain an attack.

c. **Likelihood that the policy mechanisms will cause undesirable system side effects.** Four factors could cause use of IDIP against the system: (1) false alarms causing inappropriate loss of system services, (2) adversaries masquerading as IDIP nodes, (3) adversaries compromising IDIP nodes, and (4) adversaries evoking an IDIP response that reduces the system's operational effectiveness. False alarms are controlled by configuring the IDIP agents through the policy mechanisms to use a low certainty value for specific detector alerts that experience shows are false alarms. With a low certainty value, the system can be configured to take either no response or only tracing. The IDIP cryptographic mechanisms help protect against an adversary directly using IDIP to cause denial of service, however, if the adversary compromises an IDIP node, that node could be used to deny access to legitimate users by falsely reporting attacks. A compromised IDIP node could also prevent propagation of IDIP messages, reducing the system's ability to respond. The current IDIP implementation only partially addresses the issue by providing a sub-neighborhood concept that allows the system administrator to place more vulnerable components in less trustworthy sub-neighborhoods. Reports by these components would receive more scrutiny and could have certainty and severity lowered to prevent severe responses from occurring based on reports from vulnerable devices. Future work could develop trust models to help decide when to trust these more vulnerable components.

d. **Assurance of containment.** To gain assured containment requires that the malicious node be isolated from the rest of the system. The local IDIP agents do not have this capability as the adversary may use alternate paths. However, the Discovery Coordinator's topology knowledge enables it to locate the set of IDIP agents necessary to cut off the adversary completely. An alternative strategy was also defined (but not implemented), where the IDIP response agents flood the IDIP network with **trace** messages to cover all possible alternate paths for the adversary (regardless of whether the attack was seen on the path). After all nodes have blocked, feedback from nodes closest to the attacker would cause other nodes along the same path to remove blocking rules. This provides blocking of all paths to the intruder without use of global topology information, however it does add a number of IDIP messages to the network traffic load and increases the number of short-term blocking rules that may block valid network traffic.

e. **Support for both local autonomy and global control.** Although the IDIP concept supports each IDIP node using some mix of local and global policies, the current implementation uses only a policy distributed from the Discovery Coordinator. Each policy file can be customized for the component, but this mechanism only supports a global policy set for an administrative domain. This strategy was taken to support configuration of large IDIP networks. In the IFE environment, IDIP neighborhoods were unmanageable without some central distribution mechanism.

f. **Appropriateness of the policy mechanisms to various attack scenarios.** Although IDIP policy mechanisms were not tested as desired in IFE 2.3, other testing with IDIP, plus analysis of how IDIP policy mechanisms will direct IDIP to respond indicate that the cost models can be adapted to achieve a variety of system response behaviors. The desired response is frequently to block the offending network traffic as close as possible to the source for an extended period of time. This is the response that IDIP's default policy provides for severe attacks. For less severe attacks (e.g., port scans), IDIP's default policy directs a trace response. These policies can be adjusted to cause other actions, and local devices can use component-specific responses to perform actions such as killing a host process or disabling a user account. The testing of IDIP agent policy mechanisms that did occur in the IFE 2.3 rerun did show that host-based responses that include killing user login sessions and disabling user accounts are very effective against exploits on the host.

g. **Role of each system component.** The current IDIP implementation has very specific roles defined for detection agents, response agents, and the Discovery Coordinator. These are described in more detail in Section 2. The specific roles have some overlap in that response components are usually associated with some access control mechanisms (e.g., firewall), and thus can be used to detect denied access events. Detectors may also support the response task in that detection components (e.g., host-based detectors) can often take a local response action that effectively stops the attack.

Beyond refinement of the IDIP intrusion response architecture and approach, we have developed components that are of value elsewhere. The IDIP message layer and cryptographic services are currently being used in the DARPA Common Intrusion Detection Framework (CIDF) standardization effort, as are a number of the detection components that we have integrated with IDIP. We are also contributing to the development of standards for intrusion response through CIDF.

Other accomplishments include-

- Identifying and developing varied responses, some of which have been implemented in the current prototype.
  - Trace traffic.
  - Firewall, filtering router, and host-based filtering of network traffic.
  - Disabling user accounts on hosts.
  - Disabling principal/role capabilities at object gateway.
  - Simple Network Management Protocol (SNMP)-based responses.
  - Disable Ethernet interface.
  - Attack back.

  - Change INFOCON, which changes policy.
  - Require use of security protocol (e.g., SSL) for future communication.
  - Kill process related to a user.
  - Kill connections related to a process.
  - Launch intrusion recovery.
  - Initiate use of fishbowl.
- Integration of Components for Intrusion Detection and Response.
  - COTS, public domain, and research prototype response components.
    - Gauntlet [1] (versions 3, 4, and 5).
    - Sidewinder [2].
    - Linux router [3].
    - TCP wrappers [4].
    - IP Filter for UNIX hosts [5].
    - Network Associate, Inc. (NAI) Generic Software Wrappers [6].
    - ARGuE [7].
    - Multi-Protocol Object Gateway [8].
  - DARPA prototype and COTS intrusion detection systems.
    - NetRadar [9].
    - Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) Basic Security Module (BSM) Analyzer [10].
    - EMERALD File Transfer Protocol (FTP) Analyzer [10].
    - StackGuard [11].
    - Odyssey Research Associates (ORA) Common Object Request Broker Architecture (CORBA) Immune System [12].
    - CyberCop Server [13].
    - RealSecure [14].
- Integration of management components.
    - Discovery Coordinator.
    - Cyber Command System.
    - Scotty network manager.

- Integration of Correlators.
    - Graphical Intrusion Detection System (GrIDS) [15].
    - Stanford Complex Event Processor [16].
    - Silicon Defense Corroborator.
- Common format for intrusion detection and response information.
    - Participated in development of CIDF Common Intrusion Specification Language (CISL) [17] as a common format.
- Central collection of intrusion detection information.
    - IDIP forwards intrusion detection and response reports to Discovery Coordinator for use by correlators and other reasoning components.

More detail on these accomplishments is found in Section 3.

1.4 Scope

This final technical report summarizes the Adaptive System Security Policies project results, including–

a. Summary of IDIP approach and the policy mechanisms implemented, including a response policy architecture for providing consistent response policy across an administrative domain.

b. Summary of project accomplishments and capabilities developed.

c. Description of the Integrated Feasibility Experiment configuration and the experiment lessons-learned.

d. Description of the CIDF demonstration configuration and the demonstration results.

e. Summary of the project, including lessons learned and recommended future work to better exploit this technology (including changes to IDIP to improve IDIP intrusion response capabilities).

2. IDIP DESIGN AND IMPLEMENTATION

Intruder Discovery and Isolation Protocol (IDIP) has evolved into a number of protocols.

a. IDIP message layer [18] (which is also the CIDF message layer). This layer also includes the following protocols.
    1. HELLO protocol for neighborhood management.
    2. Neighborhood key information distribution (NKID) protocol [19].
    3. IDIP authentication header ([20], [21]).
    4. IDIP encapsulating security payload ([22], [23]).

b. IDIP application layer.

Each of these protocols is largely independent of the others allowing us to modify one with minimal impact on the others. The following sections provide an overview of IDIP, followed by details on how these protocols work and the current implementation architecture. This provides context for understanding the policy mechanisms described in Section 2.4.

2.1 IDIP Overview

The IDIP application layer protocol coordinates intrusion tracking and isolation. IDIP systems are organized into IDIP *communities* (as shown in Figure 2-1). Each IDIP community is an administrative domain, with intrusion detection and response functions managed by a component called the Discovery Coordinator. Communities are further organized into IDIP *neighborhoods.* These neighborhoods are the collection of components with no other IDIP node between them. Boundary control devices are members of multiple IDIP neighborhoods.

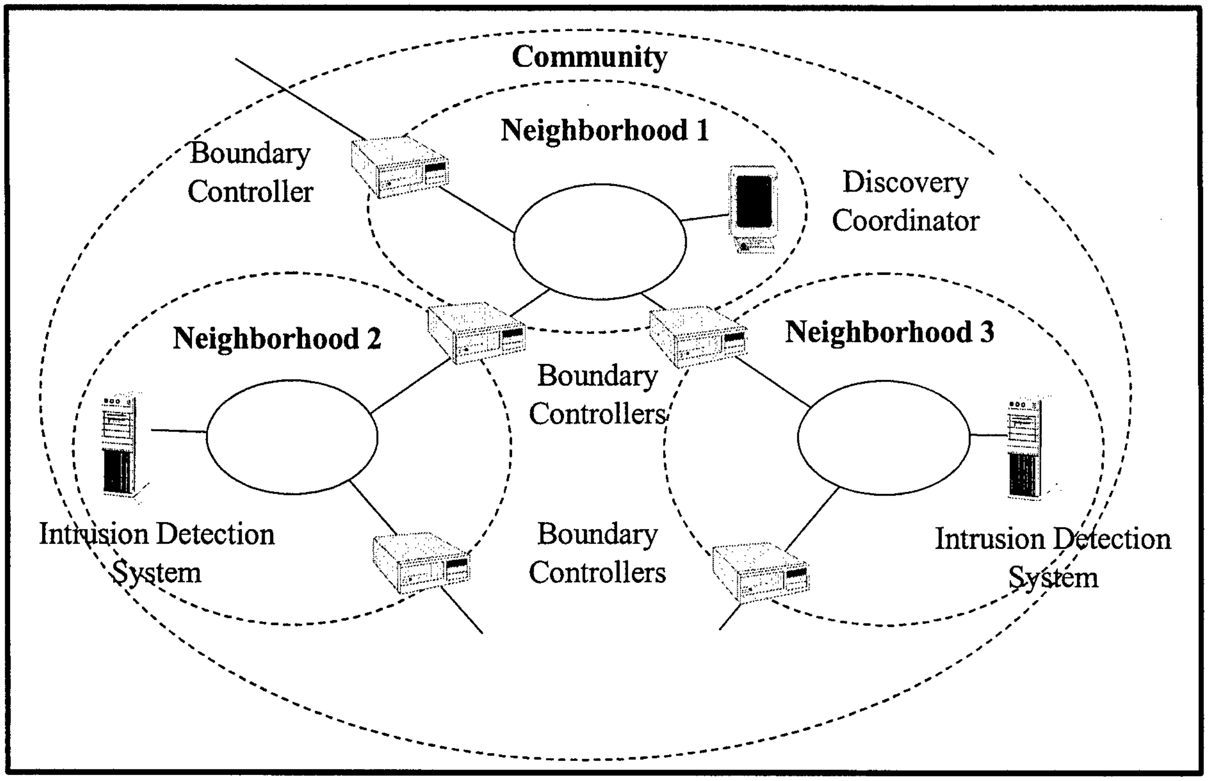

Community
Neighborhood 1
Boundary Controller
Discovery Coordinator
Neighborhood 2
Neighborhood 3
Boundary Controllers
Intrusion Detection System
Boundary Controllers
Intrusion Detection System

Figure 2-1. IDIP Communities

IDIP's objective is to share the information necessary to enable intrusion tracking and containment. Figure 2-2 through Figure 2-7 illustrate how IDIP accomplishes intrusion response. When an attack traverses an IDIP-protected network, each IDIP node along the path is responsible for auditing the connection or datagram stream[1].

---

[1] For brevity we will henceforth use the term *connection* generically to refer to both TCP connections and datagram packet streams.

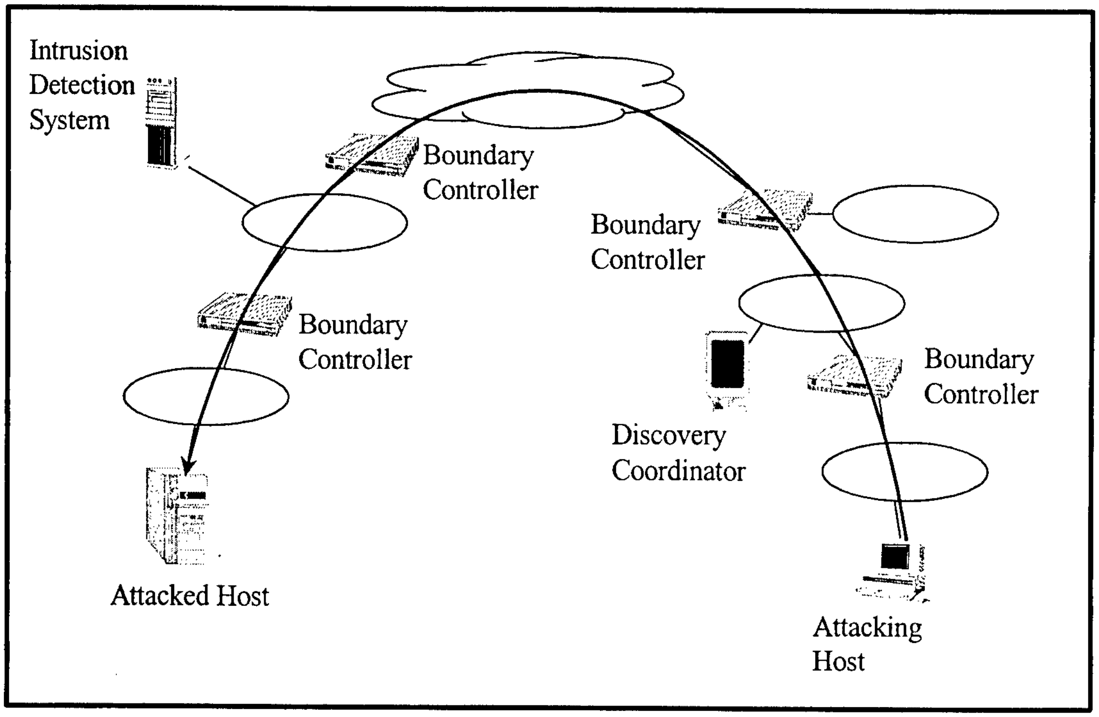
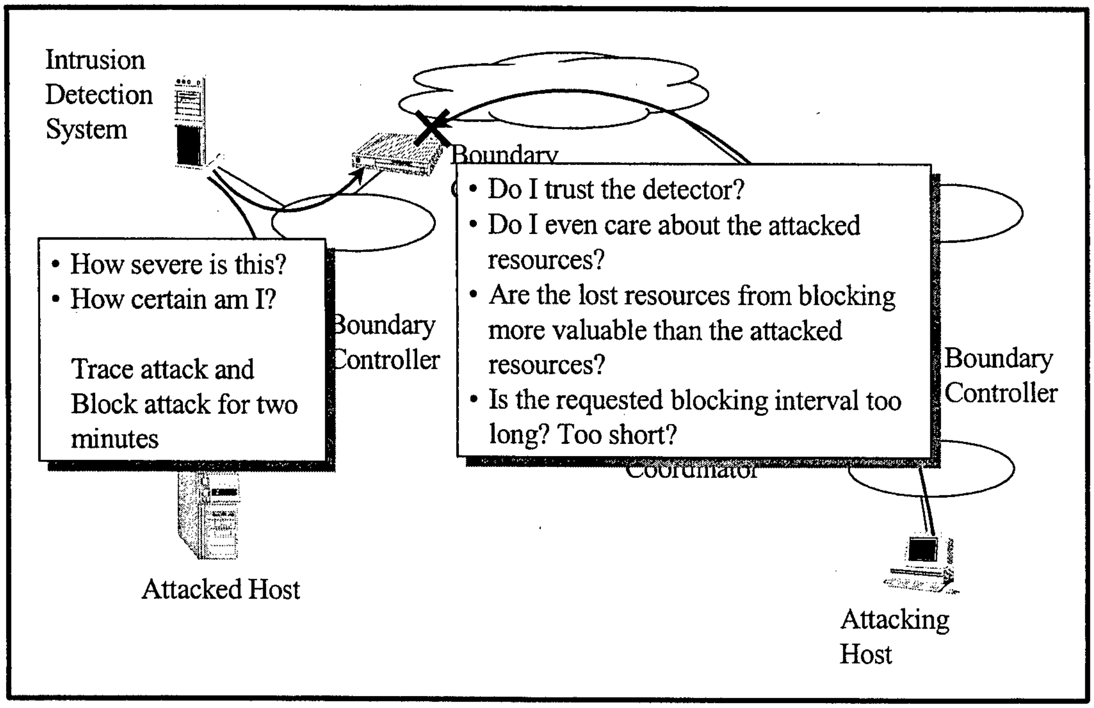

Intrusion Detection System
Boundary Controller
Boundary Controller
Boundary Controller
Boundary Controller
Discovery Coordinator
Attacked Host
Attacking Host

Figure 2-2. Attack Scenario

On detection of an attack, the detecting IDIP node determines the appropriate response, and if a response is indicated, notifies its neighbors of the attack. Each IDIP node makes a local decision as to what type of response (e.g., kill the connection, install filtering rules, disable the user account) is appropriate based on the attack type, attack certainty, attack severity relative to the type of attack and vulnerability of components under attack, what other IDIP nodes have already done, and local policy constraints (e.g., never disable http between 8 AM and 4 PM). The types of issues addressed by the response policy are shown in Figure 2-3. The attack responses are appended to the attack description prior to forwarding the attack description to neighboring IDIP nodes. This enables IDIP to trace the attack back to the edge of the IDIP-protected system, taking appropriate responses at each IDIP node along the attack path. Nodes that receive reports from neighbors determine if they are on the attack path (i.e., whether they have seen the connection described by the attack report) before forwarding the attack report. This process continues (as shown in Figure 2-4 and Figure 2-5) until the IDIP system edge is reached.

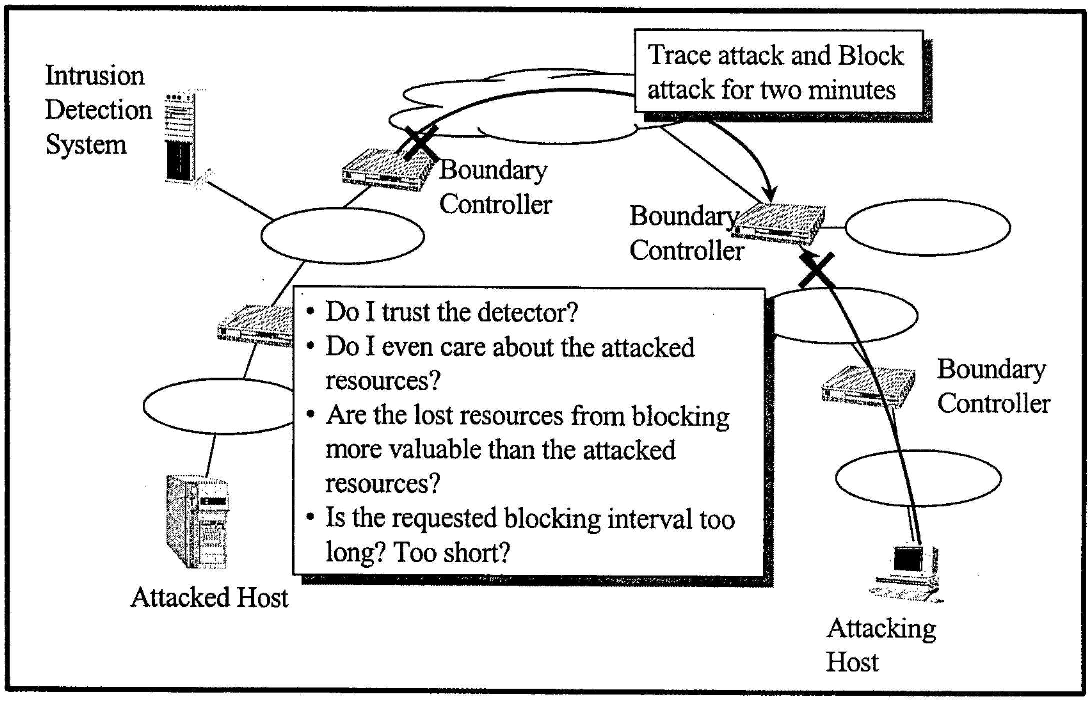

Intrusion Detection System
Boundary
Boundary Controller
Boundary Controller
Coordinator
Attacked Host
Attacking Host

- How severe is this?
- How certain am I?

Trace attack and Block attack for two minutes

- Do I trust the detector?
- Do I even care about the attacked resources?
- Are the lost resources from blocking more valuable than the attacked resources?
- Is the requested blocking interval too long? Too short?

Figure 2-3. IDIP Local Neighborhood Response


Intrusion Detection System
Boundary Controller
Boundary Controller
Boundary Controller
Attacked Host
Attacking Host

Trace attack and Block attack for two minutes

- Do I trust the detector?
- Do I even care about the attacked resources?
- Are the lost resources from blocking more valuable than the attacked resources?
- Is the requested blocking interval too long? Too short?

Figure 2-4. IDIP Remote Boundary Controller Response

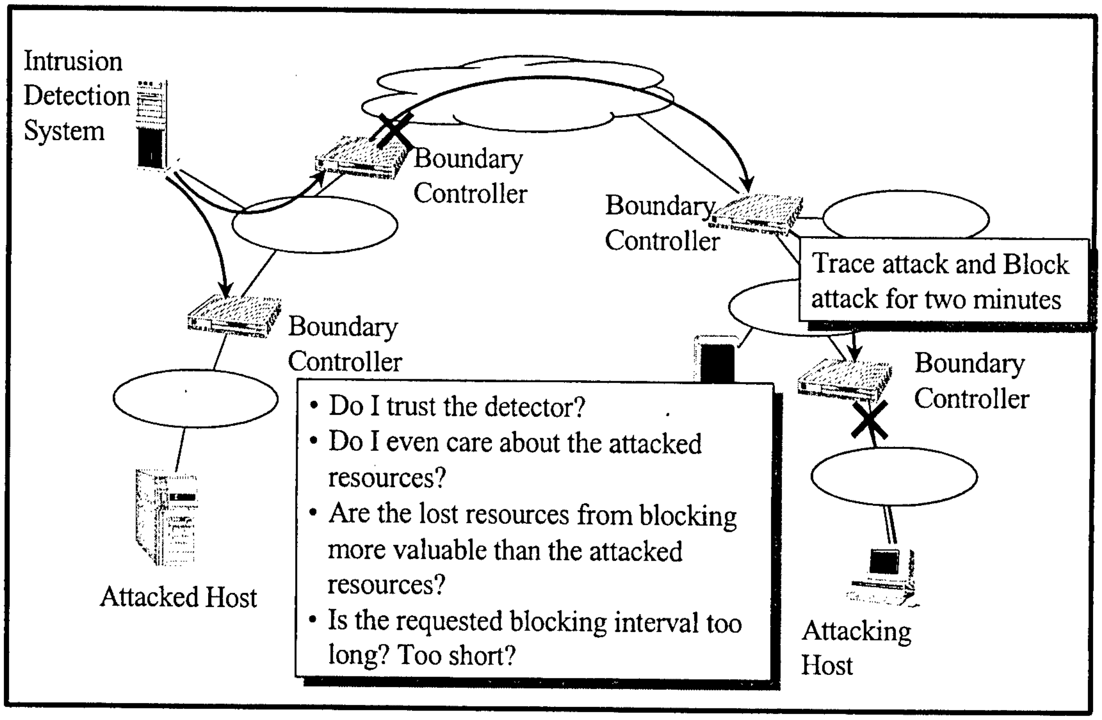

Intrusion Detection System
Boundary Controller
Boundary Controller
Trace attack and Block attack for two minutes
Boundary Controller
Boundary Controller
• Do I trust the detector?
• Do I even care about the attacked resources?
• Are the lost resources from blocking more valuable than the attacked resources?
• Is the requested blocking interval too long? Too short?
Attacked Host
Attacking Host

Figure 2-5. IDIP Remote Boundary Controller Response (Continued)

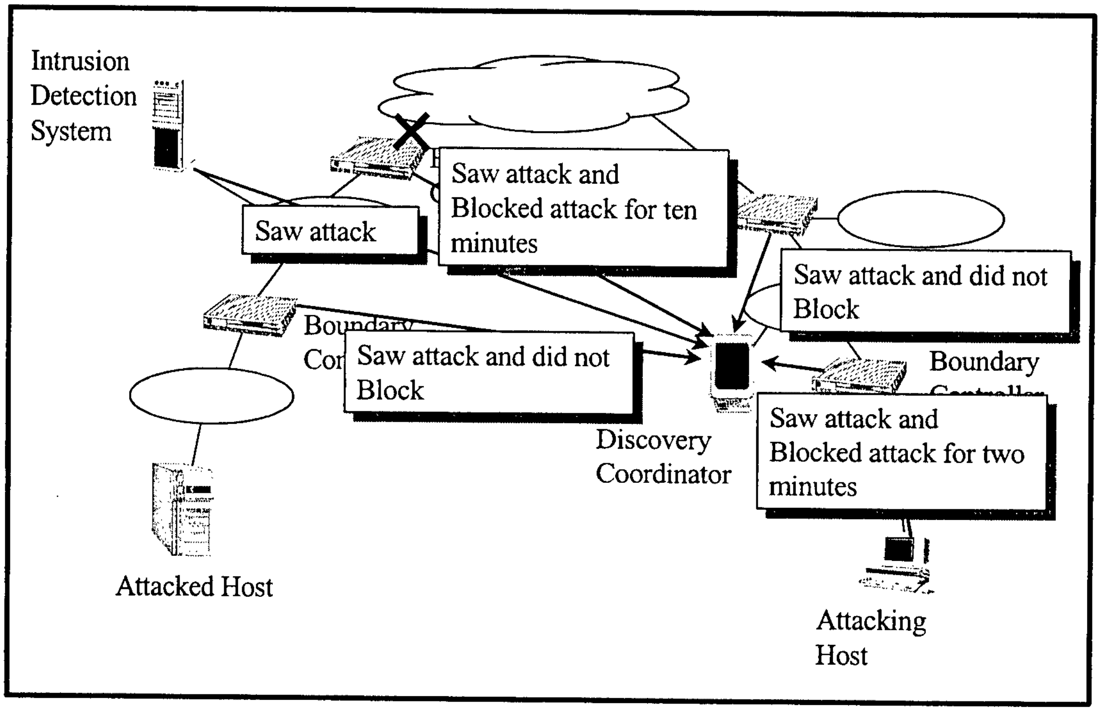
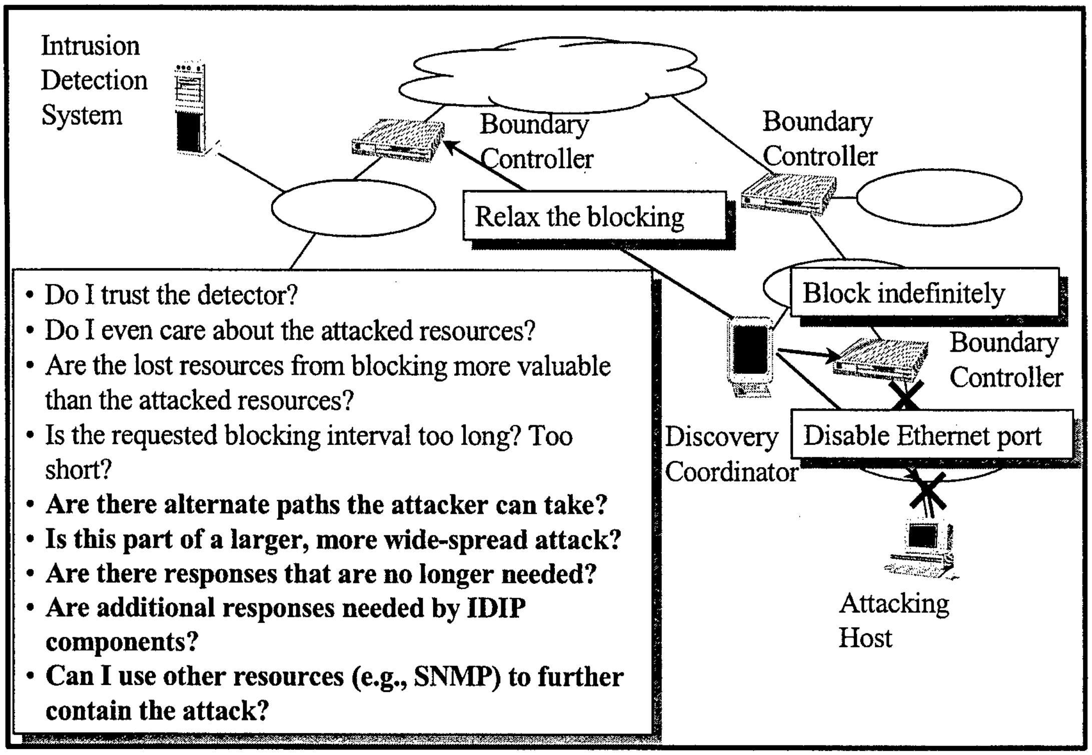

Intrusion Detection System
Saw attack and Blocked attack for ten minutes
Saw attack
Saw attack and did not Block
Saw attack and did not Block
Boundary
Saw attack and Blocked attack for two minutes
Discovery Coordinator
Attacked Host
Attacking Host

Figure 2-6. IDIP Intrusion Reporting



- Do I trust the detector?
- Do I even care about the attacked resources?
- Are the lost resources from blocking more valuable than the attacked resources?
- Is the requested blocking interval too long? Too short?
- **Are there alternate paths the attacker can take?**
- **Is this part of a larger, more wide-spread attack?**
- **Are there responses that are no longer needed?**
- **Are additional responses needed by IDIP components?**
- **Can I use other resources (e.g., SNMP) to further contain the attack?**

Figure 2-7. IDIP Discovery Coordinator Optimal Response

Additionally, each IDIP node sends a copy of the attack report (along with the local responses) to the Discovery Coordinator (Figure 2-6). The Discovery Coordinator can then correlate reports to gain a better overall picture of the situation, and also issue response directives back to individual nodes (Figure 2-7) to either remove an unnecessary response (e.g., firewall filtering rule), or add a response (e.g., firewall filtering rule along an alternate attack path). The Discovery Coordinator is expected to be co-located with the domain's network management facilities, providing the Discovery Coordinator with the network global topology, enabling the selection of the optimal points in the network to block harmful connections.

Figure 1-1 not only shows intrusion detection systems and boundary controllers as IDIP nodes, but also shows that hosts may participate in an IDIP system. Hosts can provide more fine-grained responses as they can trace the intrusion back to the process and user initiating the intrusion from the local host. When an intruder is performing an attack after hopping through multiple hosts, IDIP-enabled hosts allow the intrusion to be traced back through these hosts, which is not possible if only boundary controllers participate in the IDIP system.

Note that allowing hosts to participate in IDIP raises two significant issues for the underlying protocol mechanisms: (1) IDIP neighborhoods may grow to be very large, and (2) some IDIP nodes may be significantly less "trustworthy" than others because they may have a number of vulnerabilities available for an attacker to use. Because neighborhoods may grow very large, IDIP is designed for multicast operation. At the application level, all neighborhood communication is multicast. This second factor implies that some IDIP nodes may be compromised and potentially used against the system. For this reason, IDIP has features that enable it to distinguish less trustworthy components from more trustworthy components.

2.1.1 IDIP Application-Layer Protocol

IDIP is organized into two primary protocol layers: the IDIP application layer and the IDIP message layer. The application layer protocol accomplishes intrusion tracking and containment through three major message types: (1) **trace**, (2) **report**, and (3) Discovery Coordinator directive.

An IDIP **trace** request message is sent when an event or event sequence is detected that is determined to be sufficiently intrusive to warrant a response (which may be to trace the events or to trace and block the events). The **trace** request message includes an event description, including a description of the connection used by the intruder. Each IDIP node receiving the **trace** request uses this information to determine if the attack passed through the node. At each hop in the path, there is a possibility that this description may need to be modified due to network address translation, firewall proxies, or a user passing through a host. The protocol supports translating the attack description by appending a translation record to the end of the **trace** message. This allows tracing through hosts, firewalls, and routers. The limitation is that once a non-IDIP node that modifies the connection is reached, the connection can be traced no further. Note that the tracing mechanism is based on what the components have seen and recorded in their audit trail, rather than based on network routing tables or other dynamic network state. This approach also enables tracing of connections that spoof source addresses.

In the **trace** message, the detector specifies whether this event requires blocking in addition to tracing. Each node receiving the **trace** message is not obligated to perform the specified blocking rules, but all must perform the trace function. Local nodes can either use the suggested blocking or take some other node-specific action based on local policy. Blocking can be inserted for a limited time or until the system administrator reverses the action. When timed blocking rules are applied, the IDIP software monitors the clock to determine when to remove the blocking rule. Most responses taken by IDIP nodes are capable of being reversed. They are viewed as short-term reactions to provide system administrators time to perform whatever damage assessment and recovery actions are required. In the current implementation, the node responses to **trace** messages will typically block traffic for a short duration (e.g., a few minutes) to provide time for the Discovery Coordinator to determine an optimal response.

An IDIP **report** is simply a copy of a **trace** message that is sent to the Discovery Coordinator by each component that receives a **trace** message. This enables the Discovery Coordinator to both discover the attack path and to determine an optimal global response based on mission constraints.

To help prevent flooding the IDIP network with **trace** and **report** messages, repeated detection events are accumulated at the detector and sent as a single summary report.

Once the Discovery Coordinator has determined an optimal response, it sends directives out to nodes whose response requires altering. There are two types of Discovery Coordinator directives: (1) an **undo** message requests that the node reverse a previously taken IDIP blocking action (e.g., open up a service that was blocked at a firewall) and (2) a **do** message to take another action (e.g., extend the duration of a blocking rule). The Discovery Coordinator may request any action supported by the local response component, such as disable a user account or modify a host's policy. If the Discovery Coordinator is co-located with the system network management infrastructure, then the Discovery Coordinator can use the network management resources to take actions at non-IDIP nodes.

The Discovery Coordinator represents a single point of failure in the IDIP system, making it a target for denial of service attacks. If the Discovery Coordinator is not available for directing an optimal response, IDIP nodes can take increasingly severe responses when attacks continue following the initial response, reducing the reliance of IDIP on Discovery Coordinator actions.

To support communication between the varied IDIP nodes requires a flexible and extensible language. IDIP uses the CISL [17] developed by the CIDF working group as the language for describing attacks and responses. This language includes terms for describing the blocking actions used in the current IDIP implementation, and can be easily extended (by adding new terms) to support additional responses as they are developed. IDIP currently uses only two actions: block and allow. These can be used with various objects (e.g., users, processes, messages, or connections) to cause a number of different responses. Multiple block and allow actions can be specified in one message, each action having its own objects against which to apply the action. As an example, a "block user" message is interpreted as a request to stop that user from doing anything. A "block user and connection" message is interpreted as a request that the user be prevented from using the specified connection. Connection information includes protocol, source address, source port, destination address, and destination port. Any of these parameters may be wildcarded. Response messages can also include a specification of when to start and stop the actions.

There are several types of IDIP devices, including intrusion detection components, boundary controllers (i.e., firewalls and routers), network management (called the Discovery Coordinator), and end systems.

For IDIP, there are three terms that require special definition:

- Neighborhood – An IDIP neighborhood is a collection of adjacent IDIP nodes (i.e., two IDIP nodes are neighbors if they have no IDIP nodes between them).
- A Discovery Coordinator is an IDIP node that receives attack descriptions and descriptions of each IDIP node's response, and potentially directs the overall system response. Each IDIP node currently reports to a single Discovery Coordinator.
- A "community" is a set of IDIP neighborhoods with a common Discovery Coordinator.

Communities may be either hierarchically related or may be peers. This models the administrative environment in which IDIP nodes operate. Within a large organization, there may be a single Discovery Coordinator that serves the entire organization. Different sub-organizations within the organization may have their own Discovery Coordinators that live below the organization's Discovery Coordinator in the hierarchy. Different organizations will have peer Discovery Coordinators. Essentially, each Discovery Coordinator corresponds to an administrative domain and Discovery Coordinators have reporting relationships that follow the relationships of the corresponding administrative domains.

2.1.2 IDIP Message Layer

The IDIP message layer provides a simple, secure, reliable multicast mechanism for IDIP applications. This layer also supports unicast message transmission between IDIP nodes. The multicast functionality allows IDIP applications to communicate with all neighbors using a single Application Programmer's Interface (API) call. The message layer multicast functionality can use either the IP multicast or IP unicast service, but provides a multicast interface to the application layer.

The message layer is responsible for reliably delivering the messages to each neighbor for multicast transmission and to the single destination for unicast transmission. The message layer determines and applies the cryptographic mechanisms required for the message.

The message layer has three major protocol components outside the reliable delivery mechanisms: (1) neighborhood management, (2) key distribution, and (3) cryptographic extensions. These are summarized below.

2.1.2.1 Neighborhood Management

The IDIP neighborhood management functions support the IDIP message layer by identifying which IDIP nodes are neighbors and maintaining the state of each neighbor so that the message layer can determine which neighbors are currently operational. The IDIP message layer provides the neighborhood management functions with notification of failed transmission, and the neighborhood management functions provide the IDIP message layer with notification of added and deleted neighbors.

2.1.2.2 Key Distribution

The IDIP key distribution protocol supports the cryptographic functions used by the message layer by distributing keys used for both confidentiality and integrity. Each node generates the keys it will use for transmitting messages and is responsible for distributing these keys to neighbors. This protocol is described in detail in [19] and has changed significantly from the initial IDIP key distribution protocol. The changes were made to make the key distribution process more robust.

2.1.2.3 Cryptographic Extensions to the Message Layer

Perceived threats to IDIP include falsification of data, one component assuming the identity of another component, or eavesdropping. If a component can masquerade as an IDIP node or modify IDIP messages, there is an opportunity for both (1) disabling detection and response mechanisms or (2) severe denial of service attacks on the system through malicious manipulation of automated response mechanisms. This is no different than the threat to a system using remote management services, as these services become a good target for an adversary. Eavesdropping is a concern primarily in hiding from attackers the details of what was detected and what automated responses are being taken.

The basic requirements for IDIP cryptography include the following.

a. Efficient cryptography for messages (e.g., **trace** messages) that must be sent to each node in a neighborhood. To minimize computational overhead, encryption and generation of integrity checksums for an IDIP message should occur once for each multicast transmission to the neighborhood. This is important because an IDIP neighborhood could grow quite large. This approach results in the cost of applying cryptography to messages for very large neighborhoods requiring the same amount of time as a message going to a small neighborhood. This approach also supports use of multicast operation for IDIP neighborhoods: each **trace** message is encrypted once and either unicast or multicast to the neighborhood.

b. Support for multicast, including multicast key distribution. For efficient operation, IDIP message layer provides a multicast interface to IDIP applications. To provide cryptographic protection for multicast IDIP messages requires support from the key distribution mechanisms. That is, the key distribution mechanism must be capable of providing to the multicast group the shared keys used for encryption and generation of integrity checksums.

c. Minimal impact on IDIP message size, as each IDIP message must fit within one UDP datagram, which is 64 kilobytes.

d. Availability on multiple platforms, including Solaris™, BSD/OS™, Linux™, and Windows NT™.

e. Ease of integration.

f. Support for multiple multicast groups within a neighborhood (e.g., to segregate key sharing relationships among boundary controllers from those involving less secure hosts).

g. The number of messages for key change, due to key refresh or a change in the neighborhood membership should not noticeably affect normal IDIP message flow.

IDIP provides both privacy and authentication mechanisms. These mechanisms are modeled after IP Security (IPSec) [24], except that they provide protection above the transport layer. This enables IDIP support for cryptographic mechanisms in systems where the local infrastructure does not provide cryptographic protection.

™ Solaris is a registered trademark of Sun Microsystems, Inc. BSD/OS is a registered trademark of Berkeley Software Design, Inc. Linux is a registered trademark of Linus Torvalds. Windows NT is a registered trademark of Microsoft Corporation.

2.2 IDIP Software Architecture

The two primary objectives for the IDIP software architecture were (1) ease of integration with various components and (2) flexibility in modifying the generic component behavior for specific components. The concept supports integration of boundary controllers, network and host-based intrusion detection systems, clients, servers, and network management components.

The IDIP software was designed for portability and is currently executing on Solaris, BSDI, Linux, and Windows NT platforms. Operating system dependencies were minimized during the development and have been encapsulated in a single file.

The IDIP software components comprise IDIP backplane and IDIP applications. IDIP applications developed to date include a generic agent and various Discovery Coordinator applications. These components are described below.

2.2.1 IDIP Backplane

The IDIP backplane executes on all IDIP nodes, providing reliable, secure communication between IDIP applications. Figure 2-8 depicts the IDIP backplane showing the message layer, with neighborhood management, cryptographic key management, and cryptographic services. The message layer provides the following services.

- Reliable message delivery, including duplicate message removal.
- Multicast messaging.
- Hop-by-hop message authentication and privacy.
- Tracking of neighbor clock delta from the local clock.

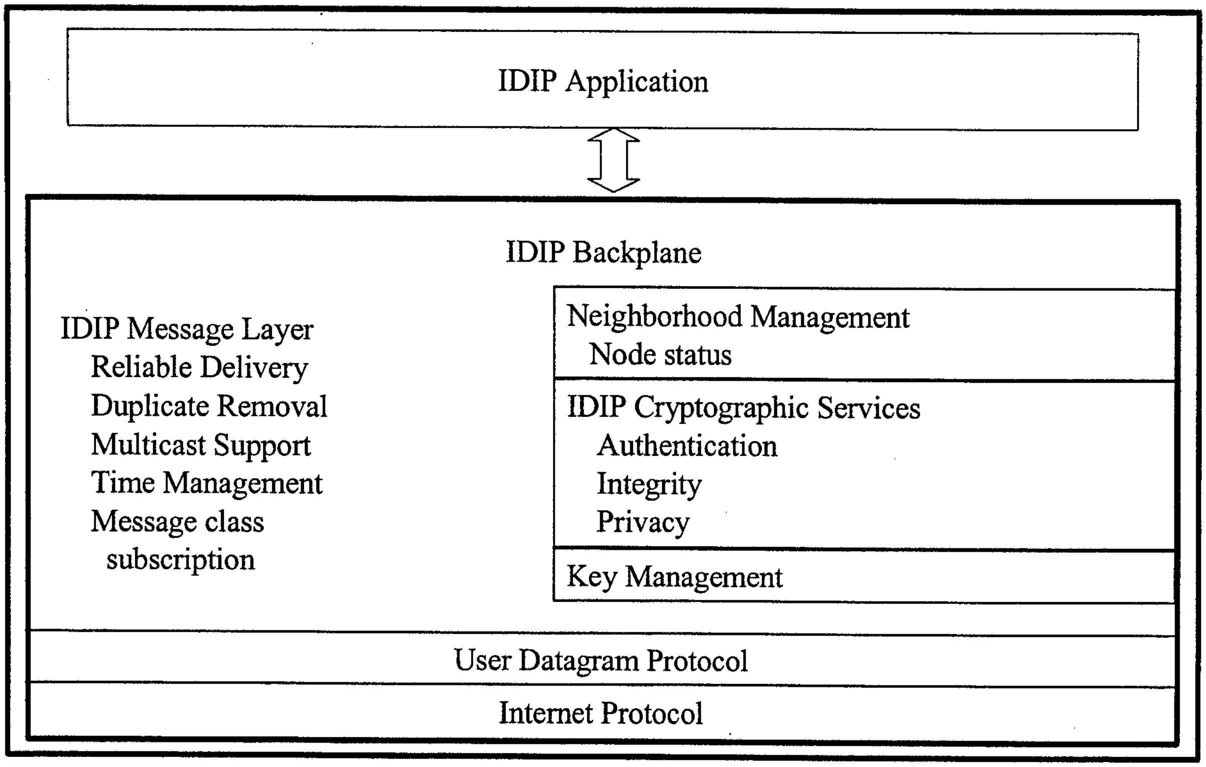

IDIP Application
IDIP Backplane
IDIP Message Layer
Reliable Delivery
Duplicate Removal
Multicast Support
Time Management
Message class subscription
Neighborhood Management
Node status
IDIP Cryptographic Services
Authentication
Integrity
Privacy
Key Management
User Datagram Protocol
Internet Protocol

Figure 2-8. IDIP Backplane Architecture

The IDIP message layer provides a socket-based interface to the application layer, enabling easy integration of new applications. An application subscribes for the message classes it needs, and the message layer delivers all messages of that class (including locally generated messages) to the application. This provides a local multicast capability that allows multiple applications on an IDIP node to share intrusion-related information.

The message layer provides reliable delivery over UDP through a simple acknowledgement mechanism. The message layer multicast functionality was designed to use IP multicast, but currently uses IP unicast services to send to each neighbor. It provides a multicast interface to the application layer regardless of whether the underlying implementation uses IP multicast or not. The time difference between neighbors is determined and adjusted using round-trip propagation delay of messages and is used by IDIP applications to adjust local time-related portions of messages, such as the time that an attack occurred. Each message has a unique message identifying number, so duplicate messages are not processed.

Neighborhood management includes maintaining status on each IDIP neighbor and forwarding that status to the Discovery Coordinator when it changes. An objective that has not yet been implemented is for this protocol to perform neighbor discovery. The implementation currently uses a list of neighbors provided by the Discovery Coordinator. The neighborhood management function provides other message layer components with notification messages when neighbors are added and deleted, detected via periodic "heartbeat" messages sent between neighbors.

During IDIP development there were no cryptographic mechanisms available that met the full set of requirements described in Section 2.1.2.3. This led to the development of a protocol for IDIP message protection modeled after IPSec, with a simple protocol for multicast key distribution. This development was facilitated through use of the open source cryptographic library from OpenSSL [25] for developing IDIP cryptographic services and key management software. This library is available for most platforms, supporting the portability requirements. Keeping the IDIP key management scheme as simple as possible reduced the number of implementation errors.

2.2.2 IDIP Applications

The IDIP applications manage the IDIP message content that is sent or delivered by the IDIP backplane. One IDIP node in a community executes the Discovery Coordinator application. All IDIP nodes execute an IDIP agent application.

2.2.2.1 IDIP Generic Agent

The IDIP generic agent application provides a framework for building component-specific detection and response engines. As shown in Figure 2-9, the generic agent provides (1) the IDIP application protocol, (2) the interface to local detection mechanisms (via the IDIP detection interface) (3) the interface to local response mechanisms, and (4) the processing of the IDIP audit data.

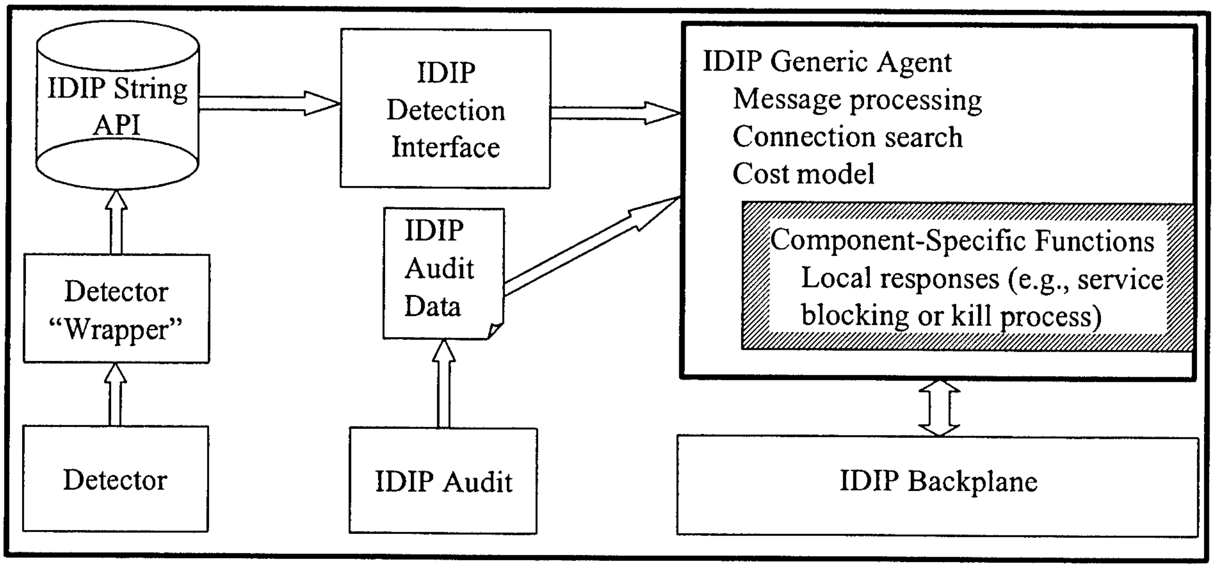

IDIP String API
IDIP Detection Interface
IDIP Generic Agent
Message processing
Connection search
Cost model
Component-Specific Functions
Local responses (e.g., service blocking or kill process)
IDIP Audit Data
Detector "Wrapper"
Detector
IDIP Audit
IDIP Backplane

Figure 2-9. IDIP Generic Agent Architecture

The generic agent software is designed to minimize the cost of new component integration. Figure 2-9 shows the IDIP agent application architecture with the component-specific routines highlighted. Figure 2-9 also shows two additional IDIP processes that support the IDIP agent.

- **IDIP detection interface.** The detection interface process provides a simple bridge from the local detection system to the IDIP agent's socket-based interface. The detection system (or a simple wrapper) writes intrusion alerts to a local file in an IDIP standard format (ASCII-formatted strings of name-value pairs). The detection interface process reads the alerts and forwards them over a socket to the generic agent. Using a file interface between the detection system and IDIP has simplified integration by allowing detection component wrappers to be developed and validated without the developer being required to install and run the IDIP software. This interface has also reduced integration debugging costs, as the file provides a good record of the messages between the detection system and IDIP. Detection systems that already produce CISL-formatted output can bypass this process by writing output directly to the IDIP backplane.

- **IDIP audit.** The audit process monitors connections to and from the local node and records this traffic in the IDIP audit data format. This software is based on a public domain package (libpcap [26]) for monitoring IP datagrams, which is available on most UNIX™ platforms. This process stores connection data in shared memory, which can be read by the generic agent. As connections end, the connection record is written to an audit file. This auditing mechanism is adequate for most IDIP needs, however, when two distinct connections represent the same data stream, additional auditing is required to connect the two data streams. For example, typically a connection through a firewall proxy will have different source ports for the connections entering and leaving the proxy. The audit process records these as two separate connections. The proxy must record that these two connections are related. For devices that perform network address translation, the address translation mechanism must record that the original and translated connections are related. Likewise, when an attacker hops through multiple hosts, they must record the relationship between the inbound and outbound connections to enable tracing the attack through the host.

The generic agent process is designed to support detection-only components (e.g., network-based detectors), response-only components, and components that perform both detection and response.

- **Detection functions.** For detection, the generic agent supports reception of detection events from intrusion detection systems, as well as other significant intrusion-related data (e.g., denied access to local host resources). For pure detection components, no component-specific functions are needed in the generic agent. IDIP **trace** messages are initiated at a node when a local intrusion detection system detects an anomaly and reports the attack to IDIP via the local detection interface process. For these locally detected attacks, the IDIP agent creates the IDIP **trace** message to send to its neighbors. The IDIP **trace** message includes an anomaly description, a value indicating how certain the detector is of this attack, a severity value based on the potential services lost from this attack, and a requested response. The agent obtains the certainty value from a detector policy configuration table. This table represents an estimate of the false positive values for each attack type. One problem encountered is that this value is highly dependent on the local environment and the detector configuration, so that it must be calibrated whenever either of these changes. The severity value is generated from a simple

---

™ UNIX is a registered trademark of X/Open Company, Ltd.

"cost model" representing the cost to the system's mission of losing the services affected by the attack. Penetration attacks are always rated a high severity as they could lead to further lost services if the penetration leads to further attacks. For the **trace** message to be sent, the severity and certainty must combine to exceed configurable threshold values. (The mechanisms for this are described in more detail in Section 2.4.5.1.1.) The IDIP agent also has a mechanism to accumulate repeated reports of the same detection events into a summary report. The first detection event is reported. Subsequent events are accumulated until either a time or event count threshold is reached, at which point the agent reports the summary event. This helps prevent a continuing attack from flooding other IDIP nodes.

- **Response functions.** For response, the generic agent executes the IDIP application layer protocol and performs local response actions. The agent receives IDIP **trace** messages from neighbors and directives from the Discovery Coordinator.
  - *Trace message processing.* For **trace** messages, generic agents use the IDIP audit data to determine if they are in the attack path. If so, the agent executes the decision logic to determine the appropriate response. The agent uses a cost model of network resource values to determine the system mission cost of taking the action requested in the **trace**. If the response action cost (in terms of lost services) is less than or equal to the attack cost (derived from the certainty and severity in the **trace** message), then the response is taken. Additional policy constraints can be placed on the response to ensure that critical services are not disabled for long periods of time unless they are already lost to the attack. (The mechanisms for this are described in more detail in Section 2.4.5.1.2.) Although the **trace** message specifies the detector's desired blocking action, the local node may perform a different action if local policy determines a better response. Most **trace**-initiated responses in the current implementation are short-lived (on the order of minutes), with the objective of providing the time needed for the Discovery Coordinator to develop a better global response. When attacks continue, however, these **trace**-initiated responses can be escalated to provide longer-term response actions.
  - *Discovery Coordinator directive message processing.* On determining the optimal system-level response, the Discovery Coordinator sends **do** messages to nodes requiring additional blocking actions, and **undo** messages to nodes whose initial responses are no longer required. Discovery Coordinator **do** messages include a specification of both "block" and "allow" rules, which can be used on objects such as connections or users. The combination of both block and allow rules in a single message enables specification of responses such as "block all network traffic except management services."

2.2.2.2 Custom IDIP Responders

The generic agent supports a flexible set of primitives that can be used to support a number of different responses. Although the generic agent provides most of the functionality required in many response components (e.g., boundary controllers), the framework allows for building component-specific response engines.

For IDIP agents that perform some response, there are two major component-specific functions required: (1) perform a blocking action, and (2) undo an IDIP blocking action. Note that blocking actions can have a different meaning for different component types. For a firewall agent, "block connection" means killing the connection and adding a filtering rule disallowing similar connections. This requires firewall-specific functions to add the appropriate filtering rule. Within a host the same "block connection" request may require reconfiguring the network services to disable a service.

Beyond these actions, a component may provide other component-specific response routines to perform more elaborate responses in specific situations. For each **trace** and Discovery Coordinator directive, the generic agent calls these routines. For example, in a system running an operating system wrapper technology [6], a suitable response might be to change wrapper policies on specific detection events. CISL provides a flexible language for specifying actions on a number of objects (e.g., process, user, message, or connection). Using CISL, new responses can be developed and carried over the IDIP system without changing the infrastructure.

2.2.2.3 Discovery Coordinator Applications

When an IDIP node sends or processes a **trace** message it sends a copy of the attack description and responses to the Discovery Coordinator in an IDIP **report** message. This enables the Discovery Coordinator to know the path of the attack and the response taken by each component along the attack path. The Discovery Coordinator also has access to other system-wide information, such as topology and component vulnerabilities. Thus the Discovery Coordinator has the information necessary to support situation understanding and generation of a system-level optimal response.

The Discovery Coordinator has a very flexible architecture allowing easy integration of new components. This is essential because cyber situation understanding and system-level course of action generation are not yet well understood. As depicted in Figure 2-10, the Discovery Coordinator can support multiple application processes to perform various system-level functions.

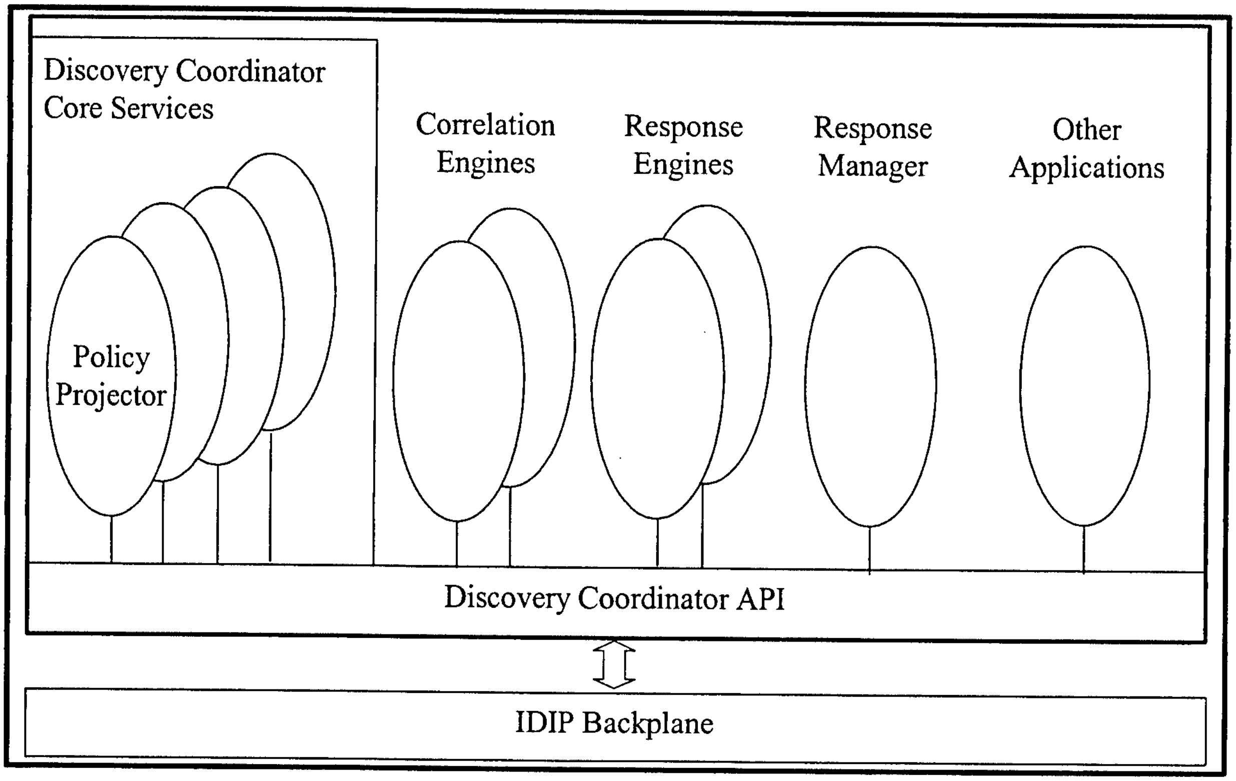

Discovery Coordinator Core Services
Policy Projector
Correlation Engines
Response Engines
Response Manager
Other Applications
Discovery Coordinator API
IDIP Backplane

Figure 2-10. Discovery Coordinator Application View

Discovery Coordinator core services include those functions that need to be shared throughout the Discovery Coordinator applications to maintain consistent system behavior.

- Data management.
- Situation display.
- Access to network management.
- Response policy management.

The response concept is for multiple response engines to propose their optimal responses and the response manager to select the response from the component best able to handle the specific situation. Although the architecture supports multiple response engines, the current implementation uses a single response engine that searches for the optimal system response based on a cost model of network resource values. The engine uses the system topology to determine all locations where a specific attack might be blocked, and then determines which location and blocking rule minimizes the overall cost to the system's mission. This cost model also reasons about user accounts with simple rules that determine when a user account should be disabled based on whether the account appears compromised.

To aid in situation understanding, multiple correlation engines can be employed. The IDIP backplane and Discovery Coordinator application programmers interface (API) allow each correlation engine to receive all attack reports from the system. These correlation engines may also produce attack reports that would be visible by other Discovery Coordinator processes. At this time, four correlation engines have been integrated into the Discovery Coordinator: (1) a simple process that attempts to combine multiple reports of the same event into a single report; (2) Graph-based Intrusion Detection System (GrIDS [7]), which combines reports based on graph algorithms to locate coordinated distributed attacks; (3) a Perl-based component developed by Silicon Defense that filters out false positives by looking for corroboration of attack reports for events known to represent false alarms; and (4) the Stanford Complex Event Processor [16].

Representing Topology in the Discovery Coordinator

The format shown in Figure 2-11 was developed for exchange of topology information within the Discovery Coordinator. It is produced by the user interface component (using the network management database) and used by the cost model.

```
(Link id
        (Attribute attribute-name attribute-value)
        ...
        (Attribute attribute-name attribute-value)
        (Node id)
        ...
        (Node id)
)
(Node id
        (Attribute attribute-name attribute-value)
        ...
        (Attribute attribute-name attribute-value)
        (Link id
                (Attribute attribute-name attribute-value)
                ...
                (Attribute attribute-name attribute-value)
        )
        ...
        (Link id
        (Attribute attribute-name attribute-value)
                ...
                (Attribute attribute-name attribute-value)
        )
)
```

Figure 2-11. Topology Format

The terms *Link*, *Node*, and *Attribute* are all keywords. The "ID" values following the *Link* and *Node* keywords are uniquely assigned numeric values (essentially used as pointers). The list of valid attribute-names and the corresponding attribute-value types can be expanded as more is learned about what component information is useful in determining optimal responses.

Initial attributes-names include *Name* and *Type*. For links, the valid Types might be "LAN", "WAN", and "Uncharted", meaning a LAN for which the Discovery Coordinator knows the topology, a WAN for which I know the topology, and a network for which I don't know the topology, respectively. For nodes, the Types might include "IDIP-Firewall", "IDIP-Router", "IDIP-Host", "IDIP-IDS", "Host", "SNMP Firewall", "SNMP-Router", "SNMP-Switch", etc., which would help give clues to the response engine as to what response capabilities the target device might have.

Under the *Link* entry of a *Node* entry one lists the link-specific node attributes (e.g., address). Other attributes could give finer grained capability descriptions if they are needed.

As an example, consider the following network configuration.

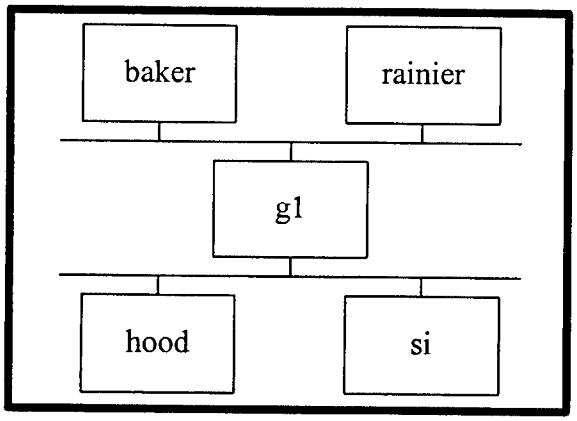


Figure 2-12. Example Network Configuration

This configuration would be described using the following message.

```
(Link 1
    (Attribute Name link1)
    (Attribute Type LAN)
    (Node 1)
    (Node 2)
    (Node 3)
)
(Link 2
    (Attribute Name link2)
    (Attribute Type LAN)
    (Node 3)
    (Node 4)
    (Node 5)
)
(Node 1
    (Attribute Name baker)
    (Attribute Type Host)
    (Link 1
        (Attribute IPAddress 134.52.160.240)
        (Attribute EnetAddress 08:00:20:71:0c:3c)
    )
)
(Node 2
    (Attribute Name rainier)
    (Attribute Type IDIP-IDS)
    (Link 1
        (Attribute IPAddress 134.52.160.244)
        (Attribute EnetAddress 00:60:08:9b:ff:bc)
    )
)
(Node 3
    (Attribute Name g1)
    (Attribute Type IDIP-Firewall)
    (Link 1
    (Attribute IPAddress 134.52.160.238)
    (Attribute EnetAddress 00:40:9e:00:01:01)
    )
    (Link 2
        (Attribute IPAddress 134.52.160.240)
        (Attribute EnetAddress 00:40:9e:00:02:0a)
    )
)
(Node 4
    (Attribute Name hood)
    (Attribute Type IDIP-IDS)
    (Link 2
        (Attribute IPAddress 134.52.160.240)
        (Attribute EnetAddress 08:00:20:22:dc:a5)
    )
)
(Node 5
    (Attribute Name si)
    (Attribute Type Host)
    (Link 2
        (Attribute IPAddress 134.52.160.240)
        (Attribute EnetAddress 00:60:08:a8:1f:a5)
    )
)
```

Figure 2-13. Example Topology Specification

2.3 Representing Intrusion Detection and Response Information

CIDF's CISL is used in IDIP to communicate **trace** and **report** information. CISL uses an S-expression syntax to form sentences describing events and responses. Figure 2-14 shows a typical CISL S-expression.

```
( And
  ( ByMeansOf
    ( Attack
      ( Initiator
        ( IPV4Address 4.22.160.163 )
      )
      ( Observer
        ( ProcessName NetworkRadar )
      )
      ( Target
        ( IPV4Address 4.22.160.140 )
      )
      ( AttackSpecifics
        ( Certainty 100 )
        ( Severity 50 )
        ( AttackID Unusual_Access Warez_server)
      )
      ( Outcome
        ( CIDFReturnCode success )
      )
      ( When
        ( BeginTime Wed Jun 16 09:07:46 1999 EDT )
        ( EndTime Wed Jun 16 09:08:46 1999 EDT )
      )
    )
    ( SendMessage
      ( Initiator
        ( TCPPort 28033 )
        ( IPV4Address 4.22.160.163 )
      )
      ( Receiver
        ( TCPPort 21 )
        ( IPV4Address 4.22.160.140 )
      )
    ( Message
        ( IPV4Protocol 6 )
        ( TCPSourcePort 28033 )
        ( TCPDestinationPort 21 )
        ( SourceIPV4Address 4.22.160.163 )
        ( DestinationIPV4Address 4.22.160.140 )
        ( ReferAs 0 )
      )
      ( When
        ( BeginTime Wed Jun 16 08:57:46 1999 EDT )
        ( EndTime Wed Jun 16 09:08:46 1999 EDT )
      )
    )
 )
  ( Do
    ( TraceMessage
      ( When
        ( BeginTime Wed Jun 16 09:07:46 1999 EDT )
        ( EndTime Wed Jun 16 09:08:46 1999 EDT )
      )
      ( Initiator
        ( IPV4Address 4.22.160.163 )
      )
      ( Message
        ( ReferTo 0 )
      )
    )
  )
  ( Do
    ( BlockMessage
      ( Message
        ( IPV4Protocol 6 )
        ( TCPDestinationPort 21 )
        ( SourceIPV4Address 4.22.160.163 )
        ( DestinationIPV4Address 4.22.160.140 )
      )
      ( When
        ( BeginTime Wed Jun 16 09:07:46 1999 EDT )
        ( EndTime Wed Jun 16 09:18:46 1999 EDT )
      )
    )
  )
)
```

Figure 2-14. CISL Example

CISL provides terms for describing an attack in terms of the time, source, target, observer, outcome, attack identity (in terms of a list of well-known attacks and attack classes), and cause. The cause of an attack (the statement beginning with "SendMessage") describes the events used to conclude that the attack occurred. Finally, a component can specify response actions that should be taken to trace or stop the attack. Response actions are currently described using three terms: trace, audit, and block. The block response is locally interpreted as whatever action is needed to stop the attack. For example, on hosts, the block response could kill the connection, temporarily disable the service, kill the offending process tree, and disable the user account to which the process belongs. On firewalls, the block response might just kill the connection and install a filtering rule to temporarily disable the service.

CISL is able to specify a wide range of sentences ranging from raw sensor events (e.g., operating system audit record) to complex attack descriptions (e.g., description of a "worm" traversing a network). Using CISL as the language for IDIP **trace** messages and attack reports enables IDIP to easily integrate with other DARPA research prototypes that are either using CISL or planning to use CISL.

CIDF Description of a GrIDS Worm

Work continues on incorporating response to distributed coordinated attacks into the Discovery Coordinator. The Discovery Coordinator's design provides for multiple attack aggregator modules to subscribe to isolated attack reports and recognize any global attack patterns. The GrIDS aggregator has the ability to detect the propagation of a malicious worm through the monitored network. Response strategies to counter worm attacks were described in previous reports. Current work focuses on describing the worm in terms that are meaningful to any type of intelligent response agent. Using the existing grammar and semantic identifiers (SID) in the CISL as a basis for this attack description, worm descriptions are built using nested combinations of the CISL *ByMeansOf* and *HelpedCause* SIDs to relate correlated worm.

TCPConnect SID events (representing worm targets with the same infecting host) are connected via *HelpedCause*, with the successful re-infection described via *ByMeansOf.* Consider, for example, a worm propagating among hosts as in Figure 2-15.

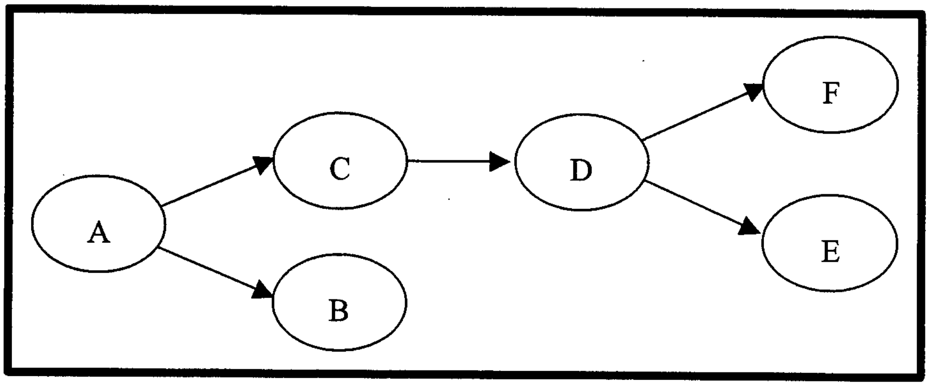


Figure 2-15. Automatically Propagating GrIDS-Style Worm

Worm connections A-B and the chain (A-C,C-D,D-E,D-F) share a common cause, namely the original infection or compromise of A itself. The sub-graph of connections (C-D,D-E,D-F) occurs by means of the A-C connection. This pattern is used to build the CISL expression shown in Figure 2-16.

```
( HelpedCause
    ( TCPConnect
        ( Initiator (Host A) )
        ( Receiver ( Host B )( TCPPort X ) )
    )
    ( ByMeansOf
        ( ByMeansOf
            ( HelpedCause
                ( TCPConnect
                    ( Initiator (IPV4Address D ) )
                    ( Receiver (IPV4Address E ) ( TCPPort X ) )
                )
                ( TCPConnect
                    ( Initiator ( IPV4Address D ) )
                    ( Receiver ( IPV4Address F ) ( TCPPort X ) )
                )
            )
            ( TCPConnect
                ( Initiator ( IPV4Address C ) )
                ( Receiver ( IPV4Address D ) ( TCPPort X ) )
            )
        )
    ( TCPConnect
        ( Initiator ( IPV4Address A ) )
        ( Receiver ( IPV4Address C ) ( TCPPort X ) )
    )
)
```

Figure 2-16. CISL-Style Worm Description

Currently the Discovery Coordinator cost model uses only the simple list of infected hosts in the attack description without making decisions based upon the shape of the worm propagation. An area of future research is to identify alternate response strategies that depend on the worm's graph structure.

For example, suppose the system detects a worm where all leaf nodes (nodes terminating in a *HelpedCause* SID) contain a heterogeneous mix of host operating systems, but nodes responsible for further propagation (nodes related by a *ByMeansOf* SID) are only hosts running Linux 5.1. It can be assumed that this release of Linux has an exploitable vulnerability responsible for the propagation. Responses to protect machines with differing operating systems aren't required, particularly considering the collateral impact they might have on normal system operation.

2.4 Policy Mechanisms

2.4.1 Requirements

Key IDIP requirements include-

a. The IDIP system must be able to respond to detected intrusions in real-time.

b. The IDIP system must support environments that span multiple administrative domains. This includes administrative domains that are hierarchical (e.g., divisions within a corporation that share a corporate master) and non-hierarchical domains (e.g., separate corporations).

c. The IDIP system must have minimal impact on system performance.

d. The IDIP system must be capable of operating while the system is under attack.

e. The IDIP system components should be capable of autonomously responding to the attack based on the IDIP message and administratively-set parameters.

To support the requirement for autonomous response, the protocol includes a detailed anomaly description and general descriptions of the desired response. That is, the **trace** messages do not indicate that the response should be to kill a process or connection, or initiate filtering rules, but rather to "block" the anomaly. The local IDIP response device uses this general response request to determine a specific set of response actions.

Note that the requirement for local autonomy does not eliminate the possibility of global control in environments where this is desired. The IDIP approach supports varying degrees of local and global control that is determined by an organization's structure and policies.

2.4.2 IDIP Nodes and Roles

As shown in Figure 1-1, the following components can participate in IDIP with the specified roles.

- **Intrusion Detection Systems.** Intrusion detection systems initiate IDIP **trace** messages on detection of an anomaly that requires system response. The detector determines the type of response required (tracing, increased monitoring, or blocking) based on the anomaly type, anomaly severity, and value of the resources under attack. The intrusion detection component may also initiate damage assessment and recovery in the local environment. When an intrusion detection component receives an IDIP **trace** message, the component determines whether the events specified in the anomaly description were seen by the component and sends an appropriate response. This provides corroborating evidence for other components in the system.

- **Boundary Controllers.** Boundary controllers provide network-based responses to IDIP **trace** messages. When boundary controllers receive **trace** messages, they determine the specific auditing and blocking actions to perform, and then pass the request to other IDIP nodes that are in the attack path. Boundary controllers also should generate their own **trace** messages when they detect an anomaly (e.g., attempted policy violation). Boundary controllers at the edge of administrative domains may also have additional requirements for modifying policy-related fields in incoming IDIP messages, and filtering outgoing IDIP messages.
- **Hosts (clients and servers).** Hosts provide fine-grained responses to IDIP **trace** messages. Like boundary controllers, when hosts receive **trace** messages, they determine the specific auditing and blocking actions that should be taken. These actions may include killing an offending process, killing a connection, disabling an account, or generating additional **trace** messages for activities related to the offending process. Hosts should also generate their own **trace** messages when they detect an anomaly (e.g., attempted policy violation) caused by a process whose parentage can be traced to a network connection.
- **Network Managers.** The network manager is the best location for the Discovery Coordinator. In IDIP, the Discovery Coordinator receives a copy of all **trace** messages sent, which includes the actions taken by each IDIP node along the attack path. Network managers are both a central location for receiving all IDIP anomaly response information and a component that can extend the IDIP response beyond the IDIP-enabled components to any component controlled by the network manager. For example, the network manager could command an Ethernet switch to disable a port even if the switch did not support IDIP. Because the Discovery Coordinator, coupled with the network manager, has both the network state information and the anomaly it can make better global decisions for intrusion response. This central component will have the information needed to assess coordinated, distributed attacks. The Discovery Coordinator then acts as an intrusion detection component, as well as a component that directs responses and recovery actions following the real-time responses triggered by the IDIP **trace** messages.

2.4.3 Parameters Used for Response Policy

Below is a summary of the parameters that may be of value in intrusion response. The summary includes the parameter's purpose and source, and how the parameter might be used. Note that not all of these are implemented in the current IDIP prototypes.

a. **Certainty.** IDIP uses certainty to represent the detector's certainty that this anomaly deserves the specified response. The detector specifies this value based on the detector's false positive rate for detecting the specified type of attack. Response components and Discovery Coordinators use certainty in determining what actions to take.

b. **Severity.** IDIP uses severity to represent the potential damage that the specified type of attack could have on the system if the attack is not stopped. The detector specifies this value based on the type of damage that the attack can cause and administrator input specifying the value of the attacked resources. Response components and Discovery Coordinators use severity in determining what actions to take.

c. **Thresholds.** Thresholds are administratively specified parameters to be used by a component in determining what actions to take.

d. **Resource values.** Resource values are administratively specified parameters that are used by detection components in computing severity and by response components and the Discovery Coordinator to determine what actions to take. Resources include potential TCP connections, User Datagram Protocol (UDP) datagrams, network capacity, host throughput, etc.

e. **Time of day constraints.** These constraints are used to provide time-sensitive resource values. Component (or Discovery Coordinator) administrative personnel generate these constraints. Detection components use these constraints in computing severity, and response components and Discovery Coordinators use these constraints to determine what actions to take.

f. **Monitoring costs.** These costs represent the system performance degradation caused by added monitoring. They are determined based on local device capabilities and administrative personnel input. All IDIP nodes use this parameter in determining whether to increase monitoring during response.

g. **Importance of remote systems.** This is a factor that can be used to lower the severity value in **trace** messages received from remote domains. Component (or Discovery Coordinator) administrative personnel specify this information to be used during response in assessing whether to take the requested actions. A second use for this value within a local domain is to reduce the value of resources as the distance (either physically or organizationally) from the resource increases.

h. **Trustworthiness of IDIP nodes.** Component (or Discovery Coordinator) administrative personnel specify this information to be used during response in assessing whether to take the requested actions. This could also be a learned value based on exhibited behavior over time. One use for this value is in specifying the reliability of a detection component.

i. **Response actions.** Detectors specify general response actions on generation of an IDIP **trace** message. Specific responses requests (e.g., create a "padded cell") could also be sent by the detector or the Discovery Coordinator. Administrative personnel should have some control over the set of desired responses. Response components use the suggested response to determine what specific local response should be performed.

j. **Recovery actions.** Recovery actions (e.g., send TCP reset to selected connections) could be generated by the detector or Discovery Coordinator and sent to IDIP response components. Administrative personnel should have some control over the set of desired recovery actions.

2.4.4 Concepts

In investigating optimal responses, the following were identified as central concepts that are further discussed in subsequent sections.

a. The IDIP mechanisms must support both hierarchical and peer relationships between IDIP communities. In attempting to respond to an attack, the organizations and their sub-organizations can have their responses centrally managed with a common view of the value of protected resources and the cost of specific responses. However, when the response crosses organizational boundaries, the view of resource values and response cost change. These different administrative domains will be reluctant to allow control from outside the domain.

b. To support real-time tracing and response that can stop the intruder before significant damage is caused, the IDIP response mechanisms that determine appropriate responses must have simple, efficient implementations in response components. If this quick response is effective at stopping the immediate attack effects, system has time to use more complex algorithms to determine a more reasoned (optimal) attack response.

c. To enable each component to determine what an appropriate response should be, we are using a "cost" model to determine which response has the best cost-benefit ratio. These cost models must be simple for the real-time response mechanisms, but can be more sophisticated for the reasoned response. The model represents the cost/benefit trade made in selecting the most appropriate response. This model depends on several factors and must balance the cost of a particular response (e.g., resources consumed and useful services disrupted) against the benefit (e.g., the intruder is stopped or slowed).

2.4.5 Cost Models

The current IDIP implementation uses cost models that assign values to resources (e.g., user accounts, access to services between components, etc.) to determine the cost of attacks and responses. Section 2.4.5.1 describes the generic IDIP agent cost models and Section 2.4.5.2 describes the Discovery Coordinator cost model. The agent cost models determine values locally, while the Discovery Coordinator cost model has the scope of the administrative domain.

2.4.5.1 Generic IDIP Agent Cost Model Implementation

Cost models provide a way to algorithmically make intrusion detection and response decisions. This section describes how the generic IDIP agents work, and suggests additional extensions or experiments that can be performed to better understand strategies associated with automated response policies.

2.4.5.1.1 Detector Agent Model

The detector IDIP agent determines both the certainty that the events represent an attack and the attack impact if it is left alone or dealt with later. These are reflected in the certainty and severity IDIP message fields. The certainty is dependent on the reliability of the sensors employed by the detector. The detector also determines an appropriate response based on the perceived cost of implementing the response. The detector's model allows use of incomplete data. For example, the detector may not know how different components along the attack path will respond and may not know much about the system-level topology.

The detector's model considers only the three "generic" responses: tracing, increased monitoring, and blocking. That is, the detector does not generally know the specific response to be taken in blocking (e.g., killing a host process, killing a proxy in a firewall, changing a filtering router's filtering rules, or disabling an Ethernet port on an Ethernet switch), and therefore cannot evaluate the different blocking options. (For components that are local to the detector, the detector may be able to determine more specific responses.) This model operates using three thresholds for tracing, monitoring, and blocking. The product of attack certainty and potential damage determine which of the three generic response actions are to be included in the IDIP **trace** or whether no action should be taken.

The policy includes records that include the INFOCON, the source IP address, the destination IP address, the service port number, the mission value or importance of the service, and the times during which this importance is to be used. IP addresses may be wild carded or prefixed with a network mask. The mission value is used to communicate to response agents the *severity* (or potential damage) of an attack.

Note that computing severity is dependent on a number of environment-specific and attack-specific parameters. These include the attacked resources' value, impact that the detected attack can have on the attacked system, and time of day. For example, the following policy could be specified.

> Denial of service attacks at midnight are tolerable, but denial of service attacks at 10 AM are not tolerable.

The initial simple model uses three thresholds: *trace*, *monitor*, and *block*. For a specific detected anomaly, the detector determines the *certainty* and attack *severity*, and from these computes the *attack value* as the product of *severity* and *certainty*. *Attack value* is used as follows.

> If *attack value* is equal to or greater than the local *trace*, *monitor*, and *block* thresholds then request the corresponding action.

The detector could optionally initiate some recovery action dependent on the type of attack. For example, the detector could reset TCP connections that were part of a SYN flood attack.

Another policy feature used in the generic detection agent is a throttling mechanism for controlling repeated reports of the same event type. When an attack is reported several times to the IDIP agent, the agent reports the first instance, and then collects subsequent instances until either a threshold count or threshold timeout occurs. At that point the agent reports a summary event describing all reported events of that type occurring since the first one. Both thresholds can be set in the policy file distributed by the Discovery Coordinator. This is particularly useful in reporting port scans. Detectors report the scans based on some threshold and keep reporting the event each time that threshold is reached. For example, in a default RealSecure policy, this threshold was set to 15 events over 30 seconds. When a subnet scan occurs, hundreds of ports are scanned across hundreds of addresses. In IFE 1.2, this created major problems by clogging up the IDIP system with thousands of reports of essentially one continuing event. This throttling mechanism addresses that problem by reporting only periodically that the scan is continuing, rather than reporting each event provided by the detector.

2.4.5.1.2 Response Agent Model

The response components must assess the certainty of attack based on (1) what the component has seen, (2) the certainty field in the IDIP message, and (3) the detector's "reliability." The cost of response is primarily based on local information. That is, the response component understands the local resources required for a response, the value of the data passing through the component through each interface (perhaps based on source and destination addresses), and the impact a response has on nodes accessible through the component. The response component also must determine how much the component should "trust" the detector and the IDIP nodes between the detector and the component. If one of the components is not trustworthy or is known to be easily penetrated, then the message validity is questionable. In this case, a decision must be based only on what the device has seen and what it can determine about the likelihood that the anomaly described in the IDIP message is an attack.

The response model parameters include factors for detectors to adjust their certainty and severity values.

The policy records for response contain the same information as those for the detection, except that a maximum and minimum response time is also present, which bounds the duration of the response.

The generic response agent is given the current INFOCON, severity, datagram description for the attacker's datagrams, and time-of-day. It searches the table top-down for the first match. The value associated with this table entry is used as the cost of response. If the cost of response is less than the severity, then the agent performs the requested block action for the time requested by the intrusion detection agent as bounded by the action duration times.

Note that these rules are parameterized by INFOCON levels to enable changes to policy by changing INFOCON. An example blocking rule is-

> For any INFOCON level,
> > if the damage exceeds 30
> > > and the protocol is 6 (TCP)
> > > and the block request source is any address (and any port range) in the 134.52.160 network
> > > and the block request destination is 134.52.160.200 within the port range 20-25
> > > and the current time is between midnight and 11:59 PM
> >
> > then perform the block request for at least 6 minutes, but no more than 50 minutes as requested by the Intrusion Detection agent.

In all cases, the response agent forwards the **trace** message.

2.4.5.2 Cost Model Implementation in the Discovery Coordinator

The Discovery Coordinator cost model, with the scope of the administrative domain, has been integrated into the Discovery Coordinator. This Discovery Coordinator cost model is implemented in CLIPS, a publicly available rule based system. Each path of mission importance, as described by an n-tuple shown below, is assigned a value ranging from 0 to 100.

> (source address range, source port range, destination address range, destination port range)

The Discovery Coordinator cost model's goal is to calculate a response that represents the least reduction in connectivity at the administrative domain level and which cuts off the attack closest to the attacker's source. While the distributed IDIP agents act as the first line of defense, taking local action based on their local cost models, the Discovery Coordinator cost model can override the decisions made locally and specify blocking instructions which can have a longer effect. The Discovery Coordinator cost model issues these override decisions as Discovery Coordinator **do** directives using CISL messages as well as SNMP commands. In addition, messages can be issued to System Management and Administration for Remote Trusted System (SMARTS) to disable specific user accounts once they have been identified as compromised.

The Discovery Coordinator sees all IDIP **trace** messages (sent as IDIP **report** messages), and therefore knows how each component along the attack path responded to the attack. In addition, if the Discovery Coordinator is co-located with the system's network manager, then the Discovery Coordinator has global topology information that can be used to determine a more optimal response. The Discovery Coordinator also can know where the administrative domain boundaries exist and can know what to expect in terms of response from neighboring administrative domains. The Discovery Coordinator can base decisions on the cost of control at each response component and can use the network management platform to perform finer grained response (e.g., through shutting down an Ethernet interface on an Ethernet switch). The Discovery Coordinator uses second-hand information and therefore must assess its reliability based on some trust model. The Discovery Coordinator also can know what response components are capable and willing to do for various types of attacks.

The Discovery Coordinator's model is much more complex than the detection and response agent models. The Discovery Coordinator can optimize response across all possible IDIP nodes, as well as non-IDIP nodes managed by the Discovery Coordinator's network manager. However, the Discovery Coordinator does not know whether any specific requested action will be completed. Individual response components may be configured to accept or not accept specific Discovery Coordinator response requests.

Additionally, the Discovery Coordinator can only affect the behavior of devices within its community. It can request Discovery Coordinators higher in the hierarchy or peer Discovery Coordinators to provide additional responses, but has little assurance that the response will actually be accomplished and may get little feedback on response effectiveness.

Given a network-based attack, there are a number of ways to respond to stop the attack, alleviate some of its effects or prevent similar attacks in the future. The IDIP system should select a response that yields the most benefit (eliminates some of the potential damage from the attack), and has the least effect upon network resources. In many cases, these two goals are conflicting. The extreme cases illustrate this: the attack can be eliminated and future attacks prevented by disconnecting all hosts; and conversely, minimal effect on network performance can be ensured by doing nothing. To provide this range of flexibility, a cost model is used that utilizes a single currency for measuring the potential damage of attacks and the strain on resources caused by responses. When an attack is detected, the Discovery Coordinator considers each possible response. If the potential attack damage is less than the cost of all applicable responses, then the Discovery Coordinator does nothing or perhaps relaxes the responses already imposed by the real-time response components. Otherwise, the Discovery Coordinator selects the response with the maximum yield (reduced attack damage minus cost of response).

Finding the optimal response entails enumerating all of the possible responses, computing the cost for each and then selecting the one with the best yield. If the number of responses is small enough, then an exhaustive search is possible and the methodology is straight forward; look at all of them and select the best one. However, if the Discovery Coordinator considers all possible response locations, the problem space becomes quite large. Even finding the minimal cut-set between the attacker and target is NP-complete. In this case, a variety of search techniques become considerations (gradient descent, heuristic based).

2.4.6 Architecture for Detection and Response Policy Projection

As currently implemented, the cost model for each IDIP agent must be configured for the local topology by the security administrator. The security administrator must take into account the subtleties of the relative importance of each mission supported by the administrative domain and the local topology. Ideally, one would like to derive all cost model inputs from one common model, thereby ensuring consistency and completeness. One would also like to be able to define detection and response policy at a level of abstraction that is both human understandable and independent of specific network topology. Topology independence is useful because this will enable dynamic responses such as reassigning IP addresses for network assets or dynamic network reconfiguration in response to an intrusion without making obsolete the detection and response policy. In addition, one would like to build one policy containing all the cost model parameters for each IDIP agent, thereby simplifying the configuration initialization process. A policy projection capability has been envisioned for the Discovery Coordinator that derives all the cost model initialization parameters discussed above from a common set of inputs. These inputs include (1) the administrative domain's topology with the associated applications allocated to each hardware resource, (2) the operational value for every mission supported by the administrative domain, and (3) detection and response behavior specification statements (also called detection and response policy commands).

Figure 2-10 shows the policy projector as a Discovery Coordinator client process. It is a peer to the cost model (a response engine) and communicates with the generic IDIP agent (to send their cost models) via the IDIP backplane. The policy projector is responsible for implementing a common model, called the network asset value model (NAVM), which values all network assets starting from a common higher level structured English language like description of importance. Figure 2-17 shows the detection and response policy commands and Figure 2-18 shows how these commands are combined with the other inputs to produce the NAVM. This section concludes with how the NAVM can be projected to each IDIP agent.

2.4.6.1 NAVM

The NAVM is fed by a number of types of information. This information includes the following information: (1) current administrative domain topology; (2) mission definition; (3) vulnerability assessment; (4) detection and response behavior policy statements; (5) dynamic detection and response thresholds; and (6) a table relating protocol usage to applications. Current administrative domain topology includes the (current) allocation of applications to hosts. Mission definition, vulnerability assessment, and detection and response behavior policy statements are represented in human readable form by the statements shown in Figure 2-17. They represent the evolving state of content for a rudimentary detection and response policy language.

Mission Value

<Operation_Name> has value <V>, where V ranges over [0..100]

Mission Asset Model

<Operation_Name> For_InfoCon (<, <=, =, >, >=) <i>
And For_DEFPOS (<, <=, =, >, >=) <j> During <Period>*
{<Requires*> (<Application>, <Application> To <Application>)}+

IDR Behavior

<Policy_Id>: For_InfoCon (<, <=, =, > ,>=) <i>
And For_DEFPOS (<, <=, =, >, >=) <j> upon_detection_of <Intrusion_Intent_Class>
with Certainty >= Y% And Severity >= <Requires*>
[Against <Application> | <Application> To <Application>]
During <Period> Perform {<Intrusion_Response_Action_Class>}+

Mission Constraints

<Constraint ID> : For_InfoCon (<, <=, =, > ,>=) <i>
And For_DEFPOS (<, <=, =, >, >=) <j> <Operation_Name> During <Period>*
(Maintain_Use_of, Preclude_Use_of)
(<Application>, <Application> To <Application>)

Vulnerability Notification

{<Vulnerability_Name> makes vulnerable <host ID> To {<CIDF Attack ID>}*}*

Trigger Command

<Trigger_Id>: For InfoCon (<, <=, =, > ,>=) <i>, upon detection of:
{<Intrusion_Intent_Class>, <Intrusion_Response_Action_Class>}
(Send_Notification_To <Application> , set Cyber_DEFPOS to <x>)

Where:

| means OR
{ }* means Null | { }+   [ ] means 0 or 1 instance of
{ }+ means 1 or more ( ) means select one of

Figure 2-17. Detection and Response Policy Components

Cyber Defense Posture. An additional detection and response behavior concept being considered to enable automatic adjustment of local administrative domain changes for detection and response behavior is a notion called cyber defense posture (DEFPOS). The trigger command shown in Figure 2-17 would automatically raise DEFPOS under specified conditions. The specific definitional statements for DEFPOS are still under investigation. DEFPOS would then be used in other policy statements to automatically change the system detection and response behavior as the system's defensive posture changes. This mechanism complements the manually adjusted INFOCON, enabling the system to provide a mix of automated and manual dynamic policy changes to varying conditions. Note that the DEFPOS mechanism can also provide another manual control if there are no trigger commands.

DEFPOS could be implemented in the Discovery Coordinator or in each IDIP agent. Implementation by each IDIP agents would add to the defensive mechanism's robustness because each agent could change its own behavior in response to specific pre-conditions.

Dynamic Mission Value. One issue with building up a NAVM from a set of static mission statements is that operational value is treated as static. In fact, operational value is dynamic and the NAVM should take that into account. This is addressed by having the system support a dynamic adjustment to the importance of various mission operations supported by the administrative domain at any given time. Incorporating these dynamic mission values into the NAVM and projecting the results to the domain's detection and response components enables the IDIP agents to better adjust detection and response mechanisms as mission profiles and resource requirements change. These dynamic mission values are represented diagrammatically as slide bars because they lend themselves to human manipulation via a GUI.

Mission asset model statements relate a mission operation to software applications using the "requires" relationship, which defines how strongly an operation requires a particular application. This relationship implements a "fuzzy logic" semantic value that includes the following range of modifiers: critically requires, definitely requires, requires, moderately requires, slightly requires, critically uses, definitely uses, uses, moderately uses, and slightly uses.

The detection and response behavior statements define specific allowable responses to intrusions detected. If the references to applications are omitted, then these statements specify administrative wide response. If the statements included specific references to applications, they can provide specific focused response for protection of critical assets. The most general form of these statements is specified to enable research to better understand the precision with which detection and response behavior can be controlled. Intrusion intent classes include the following types: probe, denial of service, unauthorized access, data corruption and disclosure. It should be noted that this represents a slight generalization of the attack types currently implemented in the IDIP protocol. Intrusion response action classes include trace, monitor/audit, notify, slowdown/delay, block for delta t, kill/terminate, and dynamic response.

The mission constraint statements provide constraints on the response system behavior and are applied after the response is generated.

Note that many of these statements incorporate the idea of INFOCON. INFOCON is a global state parameter that controls a large class of behaviors. Distinguishing detection and response behavior based on INFOCON establishes specific security posture operating regimes. INFOCON values are specified (in order of increasing severity) as normal, alpha, bravo, charlie, delta. The specific definition of INFOCON for each administrative domain is represented by the detection and response behavior that are encoded for specific values of INFOCON. The INFOCON concept has been incorporated into the generic detection and response components. The capability exists today to project INFOCON changes from the Discovery Coordinator to the set of IDIP agents.

The vulnerability notification statements identify vulnerabilities against the operating system and services running on the hardware assets in the administrative domain. The vulnerabilities are important ancillary information to detection and response policy because the level of vulnerability to particular attacks will govern the value assigned to those attacks by the policy projector. For example, if the hosts in an enclave are not vulnerable to a particular class of buffer overflow attacks then if those attacks are seen by intrusion detection systems, the intrusion response agents protecting the enclave can effectively ignore requests to block thus preserving network connectivity to (or through) the enclave.

A table relating the mission applications to their use of network protocols is also required to translate the detection and response security policy to the specific form required by the IDIP agents.

The intrusion detection and response system also has internal controls that allow for near real-time modification of the thresholds determined by the (intrusion intent class, intrusion response class) tuples in the detection and response behavior statements. As with the dynamic mission values, they are represented diagrammatically as slide bars. Until more experience is gained with dynamic control of detection and response systems, they will need to be manipulated by humans via a GUI. These will be implemented as "slide-bars" to provide an easy to understand interface for users. There is one slide-bar to modify detection: certainty. There are six slide-bars for response, one for each response class: monitor/audit, notify, slow down connection, block delta t, kill/terminate, or dynamic response. The sliders act globally in a linear fashion to adjust the response using a simple mathematical function. They have been designed to complement the use of INFOCON, which is envisioned to provide a global non-linear response to intrusion detection.

2.4.6.2 Computing the NAVM and Projecting Detection and Response Policy

The context for computation of the NAVM is shown in Figure 2-18 below. There are five major steps. The first step converts the topology independent policy statements into an internal form that takes into account the current topology. The system description includes the allocation of mission application software to hosts. Host IP addresses and service port numbers are substituted for application names in the mission asset model to yield a topology dependent representation of the information.

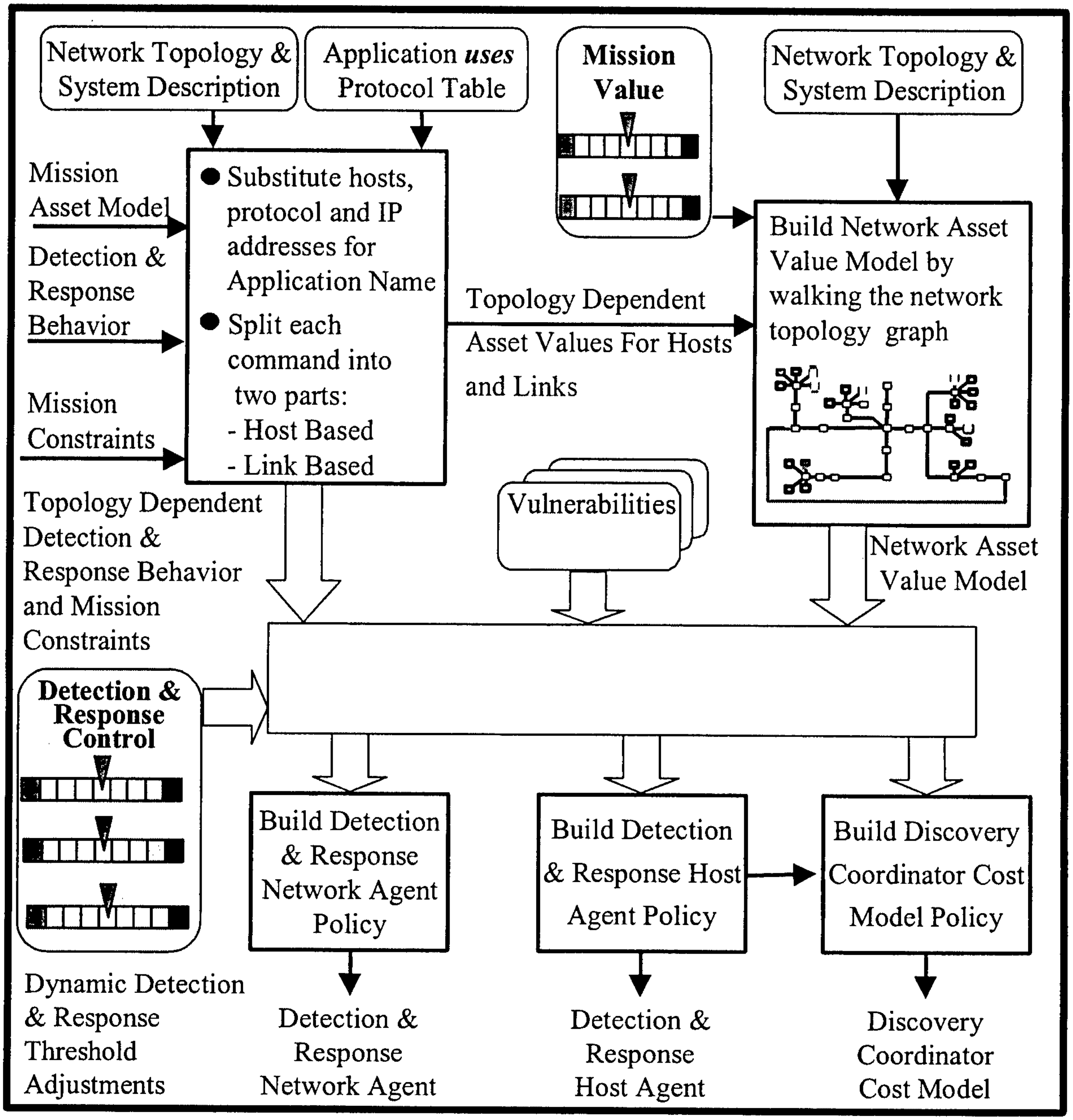

Network Topology & System Description
Application uses Protocol Table
Mission Value
Network Topology & System Description
Mission Asset Model
Detection & Response Behavior
Mission Constraints
Substitute hosts, protocol and IP addresses for Application Name
Split each command into two parts:
- Host Based
- Link Based
Topology Dependent Asset Values For Hosts and Links
Build Network Asset Value Model by walking the network topology graph
Topology Dependent Detection & Response Behavior and Mission Constraints
Vulnerabilities
Network Asset Value Model
Detection & Response Control
Build Detection & Response Network Agent Policy
Build Detection & Response Host Agent Policy
Build Discovery Coordinator Cost Model Policy
Dynamic Detection & Response Threshold Adjustments
Detection & Response Network Agent
Detection & Response Host Agent
Discovery Coordinator Cost Model

Figure 2-18. Computation of NAVM

Also during this first transformation, the mission asset model statements are split into two subclasses of statements: host oriented statements and network link oriented statements. This distinction arises because some missions may require an application that receives its inputs from and produces all its outputs for other applications on the same host. The other subclass includes applications that must communicate over the network within itself (e.g. client – server based applications) and applications that must communicate with other applications. The detection and response behavior and mission constraint statements go through a similar transformation. The application is translated into host IP, service (i.e., transport port) tuples.

The next major step is to build up a graph structure that represents each computational node in the administrative domain. First, mission value is accumulated for each host. The host's value to the network is the sum of {application strength in supporting the mission multiplied by the normalized mission value}. Application strength is obtained from the Figure 2-19 below that relates the strength of the asset requirement.

Requires Strength	Asset Value Multiplier
Critical Requirement	1
Definitely Requires	0.9
Requires	0.8
Moderately Requires	0.7
Slightly Requires	0.6
Critically uses	0.5
Definitely uses	0.4
Uses	0.3
Moderately uses	0.2
Slightly uses	0.1

Figure 2-19. Default Requirement Strength Definitions

Normalized mission value is the mission value divided by the sum of the mission values supported by the administrative domain. Normalization of mission values ensures that the range of values associated with each asset is a value from 0 to 100. This allows response thresholds to be mapped to the same scale, independent of the number of missions supported by the administrative domain.

After each host is valued, the host values are propagated to network links. This reflects that for the mission's application software to be successful in performing its part of the mission, it must receive information from and send information to other applications on the network.

The next three steps, which may be performed in any order, consist of building the inputs for the IDIP agents that can control link responses, host-based IDIP agents, and the Discovery Coordinator cost model. In each case, the value of the protected assets is derived from the NAVM. Each step results in production of a set of detection and response policy objects, which can be forwarded to the target IDIP agent. One unique detection and response policy object is built for each agent in the administrative domain.

To compute detection and response policy for a host-based IDIP agent, the policy object is created which specifies the asset value for the applications it hosts. The detection and response behavior statements are translated into specific thresholds for the various response classes that the host-based IDIP agent can perform. Response classes not supported by the host-based IDIP agent are ignored by this step in the policy projection process. To the extent possible, as supported by the host-based IDIP agent, as much precision or granularity is provided in the policy object. The goal is to provide the mission value associated with each protocol, if the host-based IDIP agent can support response on a protocol-by-protocol basis.

To compute the detection and response policy for a network-based IDIP agent, the policy object is created which specifies the mission value carried by each protocol to be serviced on each physical interface supported by the device. The detection and response behavior statements are translated into specific thresholds for the various response classes that the network-based IDIP agent can perform. Response classes not supported by the network-based IDIP agent are ignored by this step in the policy projection process.

To compute the detection and response policy for the Discovery Coordinator cost model, the policy object is created which specifies the mission value carried by each protocol to be serviced by each link in the network.

2.4.6.3 Policy projection for Rule Based IDIP agents

The above architecture discussed projection of policy for "cost model based" IDIP agents. As an alternative (and possibly an improvement) it is possible to envision rule based IDIP agents. Depending on the level of abstraction, the above architecture could be extended to project policy for rule based IDIP agents. These agents could possibly consume the topology dependent detection and response behavior policy and mission constraints directly as rules, without the associated conversion to the cost model domain. For rule-based systems that do not use value systems, additional policy statements would be required to describe the situations under which various responses should be taken.

2.4.6.4 Dynamic Policy Projection

Beyond NAVM recalculation when changes are detected in the topology or control inputs, an even more advanced feature would be to project whole new policies in response to cyber attacks. This would address the security administrator's need to rapidly deploy changes in security posture in a network environment under attack.

2.4.6.5 Dynamic Models

The models are based on a number of configuration parameters, as well as the **trace** message contents. The model configuration parameters are set through one of the following mechanisms: (1) local administrator input; (2) central administrator input (via the Discovery Coordinator); (3) local state changes indicating a need for increased security (perhaps due to a large number of anomalies); (4) directed changes from an intrusion detection component; and (5) globally requested changes in security posture from the Discovery Coordinator.

The first two mechanisms are closely related to IDIP node management. Simple control messages from the Discovery Coordinator are used to support these types of controls.

The third mechanism is device specific, and could be related to either (1) thresholds for the number of **trace** messages or anomalous events seen or (2) recognition of events that typically precede an attack (e.g., port scans). This type of mechanism is more likely to be part of an intrusion detection component or the Discovery Coordinator. Response components do not maintain sufficient state to determine when changes in configuration parameters are desirable.

Once a detection component or the Discovery Coordinator has determined that the system threat level has increased, messages are sent to increase the security alert status of other IDIP nodes. One parameter is currently available for this function: INFOCON, which is manually set. As described above, another parameter is being considered for addition to the system: DEFPOS. This mechanism is raised by either human intervention at the Discovery Coordinator or by intrusion detection components based on their analysis of changing system threats.

2.4.7 Multi-Community Policy Issues

IDIP systems that span multiple communities raise a number of additional policy issues. The following sections discuss these issues.

2.4.7.1 Inter-Community Cooperation

The two-stage response approach is also used at inter-community boundaries. The boundary controllers exchange **trace** messages to provide a simple, quick response. However, for the reasoned optimal response, the Discovery Coordinators must be willing to cooperate, which depends on organization policy. The boundary controllers at these community boundaries must know what IDIP messages may be forwarded to and accepted from the other community. These boundary controllers must also enforce any organizational sanitization rules for **trace** export and must modify the policy parameters on incoming **trace** messages. Figure 2-20 shows both the hierarchical and peer relationships.

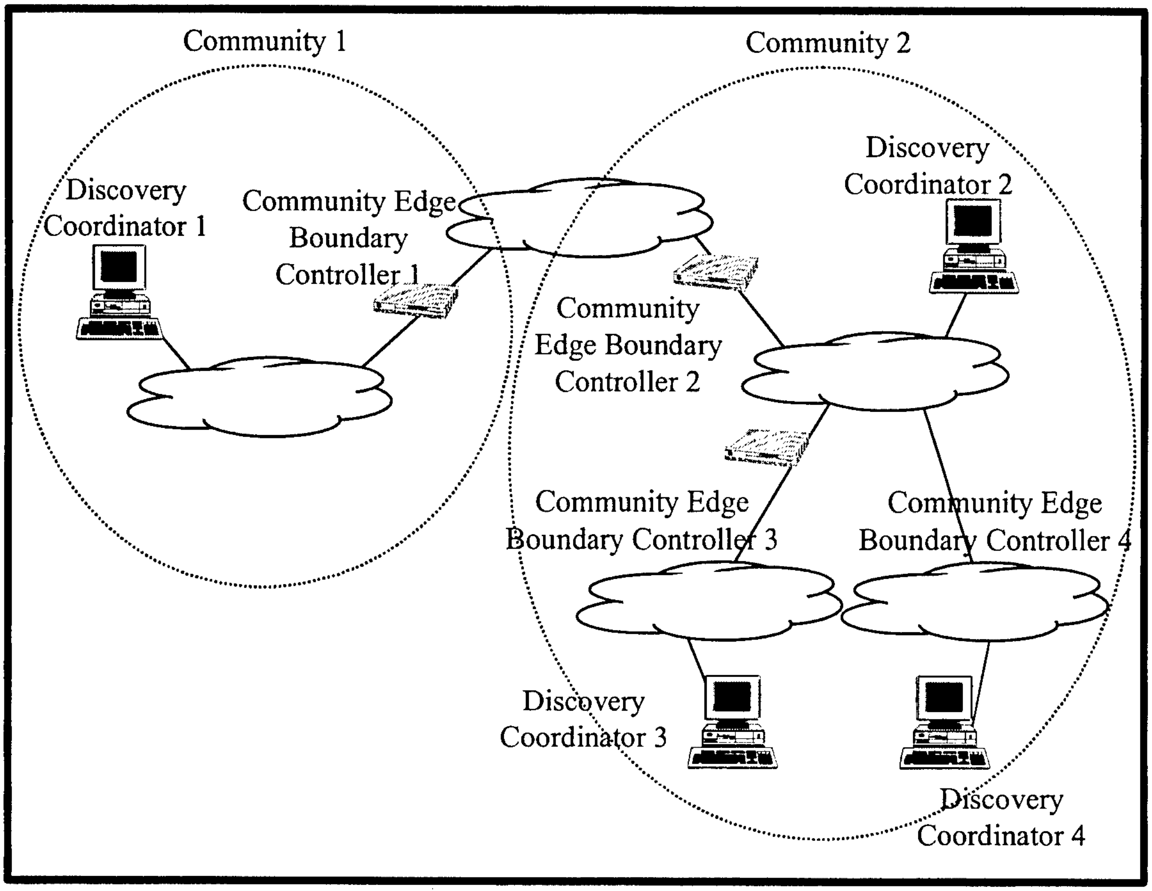

Community 1
Community 2
Discovery Coordinator 1
Community Edge Boundary Controller 1
Discovery Coordinator 2
Community Edge Boundary Controller 2
Community Edge Boundary Controller 3
Community Edge Boundary Controller 4
Discovery Coordinator 3
Discovery Coordinator 4

Figure 2-20. Community Relationships

In this figure, each cloud represents a community. Discovery Coordinators 1 and 2 are peers, while Discovery Coordinators 3 and 4 report to Discovery Coordinator 2, demonstrating a hierarchical relationship. Between hierarchically organized communities, there need only be a single boundary controller because both communities are governed by the higher-level policies. However, between peer communities, both communities will likely require their own boundary controller to enforce local import/export policies for both IDIP and user traffic. These boundary controllers (Boundary Controllers 1 and 2 in Figure 2-20) are responsible for (1) removing any data from an IDIP message that policy determines should not be exported, and (2) modifying incoming IDIP messages to reflect differences in response policies between communities. Hierarchical communities can also use this type of filtering and modification, but it seems unlikely that it is needed.

2.4.7.1.1 Outgoing Message Sanitization

In analyzing the current set of IDIP messages, the primary message that seems to require sanitization is the **trace** message. The **trace** message trailer includes both topology information and responses taken by each node on the attack path. This information may be sensitive to an organization and therefore may require sanitization at the organization boundary. One approach is to allow community edge boundary controllers to replace the response trailer with a single entry indicating the community response. Because the originator is always the first entry on the response list, this requires that the community edge boundary controller must also replace the originator identity and corresponding message identity in the IDIP header to indicate the boundary controller as the message originator. With this mechanism, to the outside world, all **trace** messages appear to be generated by the community edge boundary controller. The only other field identified for possible sanitization is severity. This field indicates how harmful the attack is. If the domain into which the IDIP **trace** is being sent is not considered fully trusted, the community edge boundary controller may set the field to a constant value to prevent foreign components that view the message from determining attack effectiveness. For domains that are fully trusted, the community edge boundary controller leaves this field unmodified.

2.4.7.1.2 Incoming IDIP Message Modification

Two fields in the IDIP **trace** header are used to support policy decisions: certainty and severity. Certainty represents how certain the originator is that this is a real attack versus a tolerable anomaly. Severity indicates how important the originator believes stopping the attack is. That is, this field represents the attack impact on the detecting organization's mission. To support inter-community relationships, two additional mechanisms are defined to reflect two concepts that arise in inter-community communication: importance and trust. The mechanisms allow the boundary IDIP nodes to modify the value of the severity and certainty fields (through the CISL addendum mechanism). Policy information provides IDIP nodes information on when to adjust these parameters based on what was attacked and what device detected the attack. This can be used at inter-community boundaries to reduce the response mechanism harshness. With this approach, IDIP nodes internal to a community do not require knowledge of whether the IDIP message came from the local community or not. Only the community edge boundary controller needs to understand community boundaries.

Importance. When sending a **trace** message to another domain, the source domain should expect that the message's "importance" will decrease because the value placed on the attacked resources generally is less. On reception of a **trace** message from another domain, boundary controllers use importance policy statements to adjust the severity value generated by the detector to change how the local domain handles the message. Note that the CISL addendum mechanism allows nodes to preserve the original value so that the local domain can determine how important responding to the attack is to the originating organization.

Trust. Trust policy statements are used to specify how confident the domain is that the information is reliable and consistent with the local domain's policy. Because several policy factors can affect how the certainty field is set, this helps a community to adjust for policy differences, as well as beliefs about the remote domain's reliability. These statements are used to modify the certainty field of the message (again through the CISL addendum mechanism) to reflect the level of trust that a node has in the detector's assessment of whether this reflects an attack. Community edge boundary controllers can use trust policy statements to control modification to certainty values to reflect that the source domain is unreliable in terms of either generating too many false positives or not adequately protecting IDIP messages.

2.4.7.1.3 Inter-Community Differences in Protection Policies

Another inter-community issue is that between different communities there may be differences in how well the IDIP message is protected from unauthorized modification. This issue is related to the trust field, above, but is sufficiently important that a hop-by-hop cryptographic indicator is being added to the IDIP **trace** trailer. These indicators specify whether the link across which the message was received provided protection compatible with the local community's protection policy. Each IDIP response component can then determine if the message has lost protection at one of the hops. It is not clear how to ensure the reliability of this mechanism unless IDIP nodes sign their response entries, which may not be feasible unless an efficient signature mechanism is developed.

2.4.7.1.4 Inter-Community Differences in Attack Definition

An issue between communities that potentially use different communication technologies is that what may appear to one domain as normal traffic, may represent a "flooding" attack to another domain. Other differences in interpreting what constitutes an attack may also be possible. The effect of this type of problem is that some communities may generate "nuisance" IDIP **trace** messages that should be ignored in the initiating community because it represents "normal" activity. IDIP has no mechanisms to directly support this issue; however, the inter-community reporting provides visibility into the problem at both communities' Discovery Coordinators. Resolution generally requires manual intervention.

2.4.7.2 Inter-Community Certification Authority (CA)

The cryptographic mechanisms rely on use of a public key infrastructure to support key distribution. Each neighborhood may have a unique CA; however, an entire organization will likely use a single CA. The CA is responsible for building the credentials used by IDIP nodes for node authentication during key exchange. The mechanisms currently do not support any specific "trust model." The use of public keys is limited to a neighborhood, so that the only requirement is that neighbors share a common CA. For inter-community neighborhoods (e.g., boundary controllers 1 and 2 from Figure 2-20, above) the two communities must agree on a common CA.

When a node is initialized, all that it requires is (1) its own private keys, (2) the CA's public key for each neighborhood to which the node belongs, and (3) where to get the node's own credentials (from disk or the address of a credential database). IDIP uses either its own credential format or X.509 certificates. In either case, the CA's signature binds the node's IP address to the node's public keys. Nodes acquire their own credentials at start-up and provide these credentials to neighbors.

Similar key exchange capabilities in support of the cryptographic protection mechanisms are required for IDIP messages that cross community boundaries. IDIP's current approach only assumes that the neighborhood must share a common CA. To preserve this feature, inter-community boundary controllers must share a common CA or have keying information (i.e., remote neighbor public keys, address, and priority) loaded manually. The current mechanisms support both modes of operation.

Currently, each physical interface device can have one CA. For interfaces to the Internet, where virtual private networks are used, IDIP cryptographic mechanisms will need to support a different CA per virtual private network.

2.5 Changes to IDIP

The primary protocol changes made to support policy mechanisms are as follows.

a. **CISL.** Use of CISL as the language for communicating IDIP **trace** and **report** messages. This allows IDIP nodes to modify the detector-generated certainty and severity fields by appending to the original message an addendum translating these fields. This allows a domain to adjust for the relative importance of resources and trust in components at different locations in the IDIP system.

b. **Response requests.** To support more efficient and consistent response, response requests were added to **trace** messages. These requests are generated by the detector and are similar to attack descriptions. They indicate what should be blocked if the component is going to filter the packet stream. For example, if the detector sees a password attack over TELNET, the attack description only specifies what was seen (TELNET), but the blocking parameters include other password-based services that should be blocked. The detector can use its knowledge of the services available in a network that are vulnerable to the detected attack type in determining the services to be blocked.

c. **Discovery Coordinator response messages.** Addition of the Discovery Coordinator messages to direct optimal responses by requesting new blocking and retracting blocking rules added by the IDIP agents in their real-time responses.

d. **Discovery Coordinator configuration messages.** Addition of Discovery Coordinator messages to push policy files out to IDIP nodes.

e. **Sub-neighborhoods.** The concept of sub-neighborhoods within a neighborhood enables configuring an IDIP system with different neighbors that may be trusted differently. That is, there may be a neighborhood that includes security devices that are "hardened" against attack, as well as clients and servers that are not. The hardened devices form a "private" sub-neighborhood within the larger neighborhood that could use different keys to separate traffic from the less trustworthy neighbors. This feature is supported by NKID.

The following additional mechanisms require further investigation to determine how the mechanism could work with IDIP.

a. **Inter-Discovery Coordinator messages.** To support both hierarchical and peer Discovery Coordinators, some messages are being added to IDIP to request responses and exchange policy information. For hierarchical Discovery Coordinators, we will also define reports from lower level Discovery Coordinators to higher level Discovery Coordinators.

b. **IDIP damage assessment requests.** To enable IDIP to be used to support damage assessment requires addition of IDIP "response" directives that include requests to trigger analysis functions (e.g., COPS) on an IDIP-enabled host. In their CRISIS project, USC-ISI is investigating this type of response mechanism. We plan to provide IDIP support for such functions and collaborate with ISI to define how this can be used. Several details require additional investigation, including the types of analysis directives (e.g., specific requests to run specific functions versus general requests to assess damage), and the amount of knowledge required by IDIP nodes to exercise this function.

2.6 Analysis

Additional response methodologies were identified and assessed, including the following.

a. Respond everywhere along the attack path, which is the same as using only local IDIP blocking.

b. Respond at locations closest to the attacker such that the attacker is completely isolated from the protected networks. This does not require use of a cost model, but does require topology information or additional messages between IDIP agents to locate the IDIP node closest to the attacker.

c. Respond at locations closest to the target such that the target is protected from all possible attack locations. The biggest problem with this strategy is that is reduces access to the target by legitimate users, and if the target is of high value to the system, will generally be a more costly response.

d. Respond only at locations that have yet to be visited by occurrences of the observed attack traffic. This strategy could potentially contain spreading, self-propagating attacks (e.g., worms). This becomes a very expensive response, but may be warranted for these types of attacks as they can have very severe impacts.

An approach for implementing b, above, that improves robustness, involves implementing these responses using only local IDIP information at boundary devices and the information from their immediate neighbors. This makes the system less vulnerable to Discovery Coordinator loss. For each case listed above, we have identified a state-machine that, when applied to local IDIP controllers, will produce the required global optimized response without recourse to decisions made by a centralized response selector.

The mechanisms implemented in the current IDIP prototypes enable IDIP nodes to more effectively respond to intrusions.

Performance Impact. The original IDIP definition minimized system performance impact by using very few resources except when responding to a suspected attack. The additional mechanisms developed add very little additional overhead. Changes from the original IDIP are described in Section 2.5. In each case, these are relatively lightweight infrequently sent messages.

Topology limitations. Because the Discovery Coordinator has a global view of system topology, there should be few limitations in responding to attacks regardless of system topology.

Undesirable side effects. IDIP-based network responses can cause useful traffic to be blocked. The cost models minimize the system-level impact of this blocking. If the cost in terms of blocked useful communication is higher than the benefit of stopping the attack, then the cost models cause the IDIP response to not block the useful traffic. The two-staged response approach also reduces the impact on useful traffic. The real-time response may temporarily eliminate useful communication, but it will be quickly replaced by the longer-term response. This minimizes the duration of the undesirable side effects.

Assurance of containment. The Discovery Coordinator integration with network management enables the Discovery Coordinator to use topology information, which is critical to ensuring that an attack is contained. New IDIP messages were added allowing the Discovery Coordinator to direct responses by response components. These additional messages enable the IDIP system to better contain the attacks once their source is located. The use of the Discovery Coordinator information for system-level intrusion detection also allows IDIP to attempt containment of distributed attacks (e.g., worms).

Local versus global control. The Discovery Coordinator provides global control for its community, but each IDIP node is also allowed to deviate from global policy when the local policy takes precedence.

Appropriateness of policy mechanisms to specific attacks. The IDIP approach is to use data-driven models that can be adjusted to modify the response behaviors to adapt to system requirements. This allows a system to tailor the response policy mechanisms to what is determined to be appropriate by an organization. Beyond this tailoring, the two-staged response approach provides a more appropriate response than the original IDIP. The general responses described in Section 1.2 can be selectively enabled depending on the attacked resource's value and the attack severity. More specific responses can also be requested by the Discovery Coordinator as the second stage of the response to provide finer-grained response.

Role of each component. The various component roles were described in Section 2.

3. PROJECT ACCOMPLISHMENTS

The following sections detail the project accomplishments, capabilities developed, and lessons learned.

3.1 Overall Accomplishments

The primary project accomplishment was developing, and gaining experience with, initial response policy mechanisms. These are described in detail in Section 2.

Although not part of the original project plan, this project contributed significantly to the CIDF effort, including (1) developing approaches and CISL constructs for communicating response requests and actions; (2) developing an approach to modifying CISL messages without losing the original data or signature (e.g., because of translation required from connections that pass through firewall proxies and network address translation devices); (3) developing the CIDF message layer; (4) contributing to CISL binary encoding rules; (5) contributing to removal of CISL specification ambiguities; (6) developing a CIDF implementation of IDIP for use in integrating components for the CIDF demonstration; and (7) serving as integrator for the CIDF demonstration.

We developed a number of new capabilities for IDIP and integrated with a number of new detection and response components. New capabilities are described in Section 3.2, and the new components integrated with IDIP are listed in Section 1.3.

Although we did not develop mechanisms for multi-community intrusion tracking and containment, we did identify several issues and some strategies for addressing these issues. This is discussed in detail in Section 2.4.7.

3.2 Capabilities Developed

New capabilities developed during this project (in collaboration with the Automated Response to Intrusion project) include-

- Standard IDIP string API to support detection component integration.
- Development of generic detection and response agent to provide most of the IDIP functionality needed for automated intrusion detection and response.
- Integration with several detection and response components to provide new detection capabilities and response mechanisms (e.g., SNMP-based responses via the Scotty network management system).
- Development of core Discovery Coordinator components to support situation assessment and generation of global responses.
- Development of cost model components for the IDIP detection and response agents and for the Discovery Coordinator.

- Development of Discovery Coordinator cost model mechanisms to maintain a list of responses that are currently in effect from previous attacks so that additional attack reports received calling for the same response can be ignored. This suppresses redundant requests issued by the Discovery Coordinator in response to similar attack reports, or from multiple reports of the same attack coming from different intrusion detection systems elsewhere in the system.
- Development of a CIDF implementation for IDIP **trace** messages and detection reports.
- Improvements to IDIP cryptographic mechanisms to improve system robustness, including use of (1) higher performing (and more readily accessible) software cryptographic algorithms, (2) more robust key distribution algorithms, and (3) X.509 certificates as IDIP credentials.
- Development of Discovery Coordinator architecture that supports abstract policy specification and distribution of policy to IDIP agents.

4. SYSTEM-LEVEL DEMONSTRATION

For IFE 2.3, we integrated several technologies with IDIP. This IFE employed a red team from Sandia to develop and execute attacks on the system and its mission. During this IFE, the system configuration was such that the red team could easily capture their flags without detection by becoming insiders. Unfortunately, without detected events, we were unable to assess the effectiveness of the responses, as there were none. Following the first run of the experiment, it was decided to leave the configuration as it was and develop new detection mechanisms. Effectively, IFE 2.3 became an experiment on detecting and responding to insider attacks. For the IFE 2.3 rerun, we developed some new detection capabilities and integrated some newly developed response mechanisms. Unfortunately, several configuration control and policy decisions prevented the rerun from being as effective as it could have been. However, some information on response effectiveness was gleaned from this set of experiments.

The following sections describe the experiment in more detail. Section 4.1 describes the experiment configuration and scenario; Section 4.2 describes the first run of the experiment; and Section 4.3 describes the experiment rerun.

4.1 Experiment Configuration and Scenario

Figure 4-1 shows the IFE configuration. In this configuration, the red team (*The Scones*) was positioned on the *Internet Cloud* with the objective of modifying orders prior to the orders being sent to Duracell. The orders are sent to the Oracle database via web-based transactions from either *Fort Lewis* or *The Preserves*. Periodically, a process on the Oracle/web server would process the orders and FTP them to Filemaker Pro. A process on the Filemaker Pro machine would periodically issue a CORBA transaction to Duracell to place the order. Web-based orders were launched from programs running on NT clients. Finally, periodic TELNET was scripted from the consultant home office to the Oracle/web server, where the script would login to the server and login to Oracle. To make the scenario somewhat realistic, the traffic generators generated web and email traffic across the *Internet Cloud*. We were able to expand the experiment's topology complexity using the Information Assurance Program's virtual private network and Boeing Security Technology Lab. Both web-based order clients and a traffic generator were run from the Boeing lab.

In Figure 4-1, the components running IDIP are italicized. These formed a single IDIP community controlled by the Discovery Coordinator in the *Regional Network Operations and Security Center*. The plan was to have three *runs* in the experiment: (1) detection only, (2) detection with IDIP real-time response, and (3) detection with both IDIP real-time response and Discovery Coordinator directed response. A fourth *run* had been planned using the Cyber Command System to generate additional responses; however, this was removed from the plan due to time limitations. If the Cyber Command System had participated in the experiment, it would have been fed intrusion-related information through the Discovery Coordinator's Cyber Command System interface software. That interface software is also capable of taking response direction from the Cyber Command System.

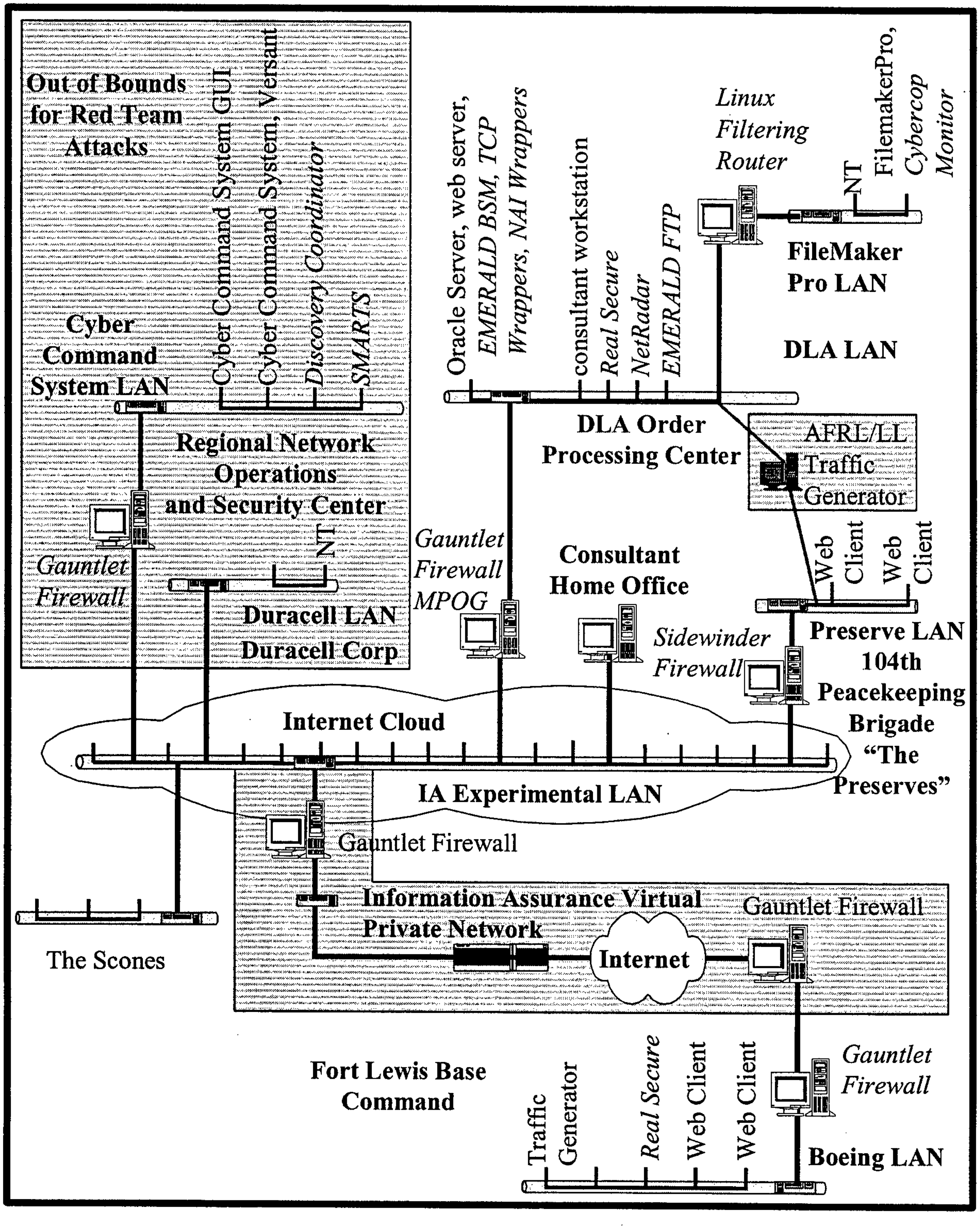

Out of Bounds for Red Team Attacks
Cyber Command System LAN
Cyber Command System GUI
Cyber Command System, Versant
Discovery Coordinator
SMARTS
Regional Network Operations and Security Center
Gauntlet Firewall
NT
Duracell LAN Duracell Corp
Oracle Server, web server, EMERALD BSM, TCP Wrappers, NAI Wrappers
consultant workstation
Real Secure
NetRadar
EMERALD FTP
Linux Filtering Router
NT
FilemakerPro, Cybercop Monitor
FileMaker Pro LAN
DLA LAN
DLA Order Processing Center
AFRL/LL Traffic Generator
Gauntlet Firewall MPOG
Consultant Home Office
Sidewinder Firewall
Web Client
Web Client
Preserve LAN 104th Peacekeeping Brigade "The Preserves"
Internet Cloud
IA Experimental LAN
Gauntlet Firewall
The Scones
Information Assurance Virtual Private Network
Internet
Gauntlet Firewall
Fort Lewis Base Command
Traffic Generator
Real Secure
Web Client
Web Client
Gauntlet Firewall
Boeing LAN

Figure 4-1. IFE 2.3 Configuration

The red team flag (i.e., objective) was to subtly modify critical order information without being caught or stopped. This includes modifying orders submitted to Oracle/web server (including on the wire from client to server, on database entry, or in the database); orders being transferred to the Filemaker Pro from the Oracle/web server (including on the wire from Oracle/web server to Filemaker Pro and files in the server containing Filemaker Pro); and orders submitted (via CORBA) to Duracell.

4.2 Initial IFE 2.3

In preparation for IFE 2.3, we integrated several intrusion detection and response capabilities with our IDIP software. Many of these had been previously integrated, but because of version changes and new features, they required re-integration. Execution of the IFE involved configuring and operating intrusion detection and response components and experiment support components (e.g., traffic generators) in both the DARPA Technology Integration Center and the Boeing Security Technology Lab. Once the components were configured, the red team performed their attacks while we collected data.

The following sections describe the integration for, and execution of, IFE 2.3.

4.2.1 Integration

For the initial IFE 2.3 run, we integrated the following technologies with IDIP: EMERALD BSM Monitor, EMERALD FTP Monitor, NetRadar, RealSecure, CyberCop Monitor, Gauntlet 5.0, Sidewinder, Multi-Protocol Object Gateway (MPOG), IP Filter, NAI Generic Software Wrappers, TCP Wrappers, and Discovery Coordinator. In addition to IDIP-directed responses, the Discovery Coordinator is able to use SNMP to disable Ethernet ports on either hosts or switches and SMARTS to disable user accounts (via Cyber Command System Versant Object Database). The Discovery Coordinator also provides attack reports to the Cyber Command System (formerly the Security Service Desk) to support human-directed response.

In support of this experiment, we developed a number of new IDIP capabilities, including (1) enhancements to the IDIP string API to support additional attack report information including information for host-based detection and response (e.g., user ID and process ID); and (2) support for customized component-specific responses. These new capabilities, plus integration with some components is described in Sections 4.2.1.1-4.2.1.6.

4.2.1.1 User-Based Responses

When an EMERALD BSM report indicates that a user account may have been compromised, IDIP provides the Discovery Coordinator cost model the user identity. The cost model generates both (1) an IDIP message to the NAI Generic Wrappers to use the "stop user" wrapper effectively blocking all processes for that user from making progress; and (2) a message to the Object Database, triggering SMARTS to disable the user account. The combination of the two mechanisms provides both an immediate response that stops current user processes, and a longer-term response that disables further account use.

4.2.1.2 Component-Specific Responses

Beyond the work to make IDIP more robust and the addition of new responses and cost model logic to use these responses, we added mechanisms in the IDIP agents to support addition of component-specific responses. The base IDIP code has been focused on firewall-like responses (e.g., blocking a service or access from a remote device). This works well for firewalls and routers, and for host-based network access control mechanisms such as IP Filter and TCP Wrappers. However, this does not support performing additional responses such as killing a user process or disabling a user account. By the start of IFE 2.3, we had developed an interface to support additional responses; we had not yet integrated additional responses as agent local responses.

4.2.1.3 MPOG Responses

A number of MPOG responses have been defined including the following.

- Deny role, which temporally disables a user role.
- Deny principal, which temporally disables use of certificates associated with the principal identity.
- Deny principal in role, which temporally disables the specified principal from using a user role.
- Deny client, which temporally disables access from the client.
- Deny principal from client, which temporally disables access from the client for connections associated with the specified principal.
- Deny service, which temporally disables new connection to a service (TCP port number).
- Deny service from client, which temporally disables new connection to a service (TCP port number) from the client.
- Deny server, which temporally disables access to the server.
- Deny service on server, which temporally disables new connection to a service (TCP port number) on the specified server.
- Require use of security protocol, which requires use of SSL for all new connections.

Not all of these are currently available, but we have defined the IDIP mechanisms necessary to support these. Some modification is still required for MPOG to support all of these.

4.2.1.4 IP Filter and TCP Wrappers

We completed integration of IDIP with the public domain TCP Wrappers and IP Filter software. Both TCP Wrappers and IP Filter work on several UNIX platforms, providing access control mechanisms on inbound network services. On receiving IDIP **trace** messages and Discovery Coordinator directives, the integration software written for these components writes filtering rules for the component-specific filtering mechanism. For TCP Wrappers, this is simply writing the rule to a file. For IP Filter, the integration software writes the filtering rules to IP Filter's memory-based filtering rule set.

4.2.1.5 Discovery Coordinator Cost Model

The major change to the Discovery Coordinator cost model resulting from this IFE was the introduction of the notion of user accounts into the cost model. The cost model now associates value with each user account, and can decide when the potential damage from a potentially compromised user account exceeds the account value. In that case, the cost model generates a disable user request, which can either be sent to the Cyber Command System or out to a specific IDIP agent. The cost model also added the capability to use some component-specific mechanisms to provide additional responses. Two such mechanisms are (1) SNMP as a response to completely disable an Ethernet interface and (2) MPOG-specific responses.

4.2.1.6 Cyber Command System Integration

The interface to the Information Assurance Program's Cyber Command System is through the Versant Object Database. The Discovery Coordinator provides attack report objects through the database to other database clients. In addition, the Discovery Coordinator can accept from the database commands to perform blocking actions or to undo IDIP actions. This allows other reasoning engines (or the user at the Cyber Command System GUI) to affect responses. The database is also the interface that the Discovery Coordinator uses to request that SMARTS disable a user account. The Discovery Coordinator provides SMARTS the user ID and host address, and SMARTS is responsible for mapping that identity to all of the user's accounts across the system, and disabling those accounts.

4.2.2 Experiment Execution

The experiment execution used multiple attacks by the red team, using alternate attack steps in each *run*. The objective of the blue team (Information Assurance component developers and Technology Integration Center staff) was to gather data from the detectors and response components to help assess the effectiveness of automated response. As mentioned previously, during IFE 2.3, no detections were made, so no data was collected.

4.2.2.1 Red Team Attack

In preparation for the IFE, the red team developed attack trees, which describe the steps required to achieve their goals. The trees include alternate paths to the objective, with each path requiring one or more steps. The red team is allowed to employ any technique within the ground rules for the experiment, and so is not limited to using publicly available exploits. The various paths are then assessed for difficulty, and the easiest paths are tried first. For this experiment, the red team used several paths on different *runs*. Each *run* has the red team starting on the *Internet Cloud* and working their way toward the objective.

Figure 4-2 through Figure 4-5 show the primary red-team attack trees that were used both in IFE 2.3 and the IFE rerun. As can be seen in Figure 4-3 through Figure 4-5, the red team was highly dependent on gaining login access to the Oracle/web server. From there, no further exploits are required to capture their flag. By providing the red team the user identity and password for the Oracle/web server by having the consultant TELNET to the Oracle/web server and login into the server and Oracle, no exploits were required for the red team to capture their flag. Without exploits being used, the detectors employed were not capable of detecting red team activities. The only red team action that was detectable by standard detectors was the network mapping activity shown in Figure 4-2 used to start the attack. Unfortunately, no detector was placed on the Internet Cloud, so that one event was also not detected.

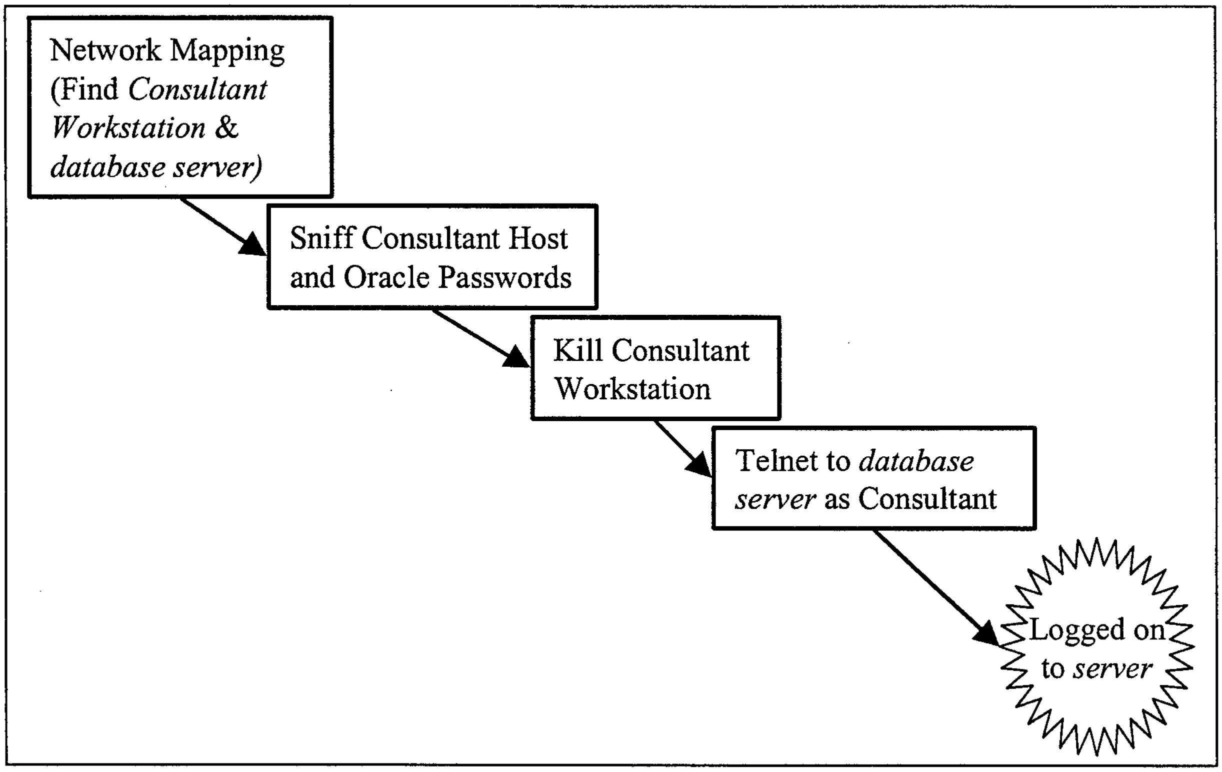

Network Mapping (Find *Consultant Workstation* & *database server)*
Sniff Consultant Host and Oracle Passwords
Kill Consultant Workstation
Telnet to *database server* as Consultant
Logged on to *server*

Figure 4-2. IFE 2.3 Attack Tree

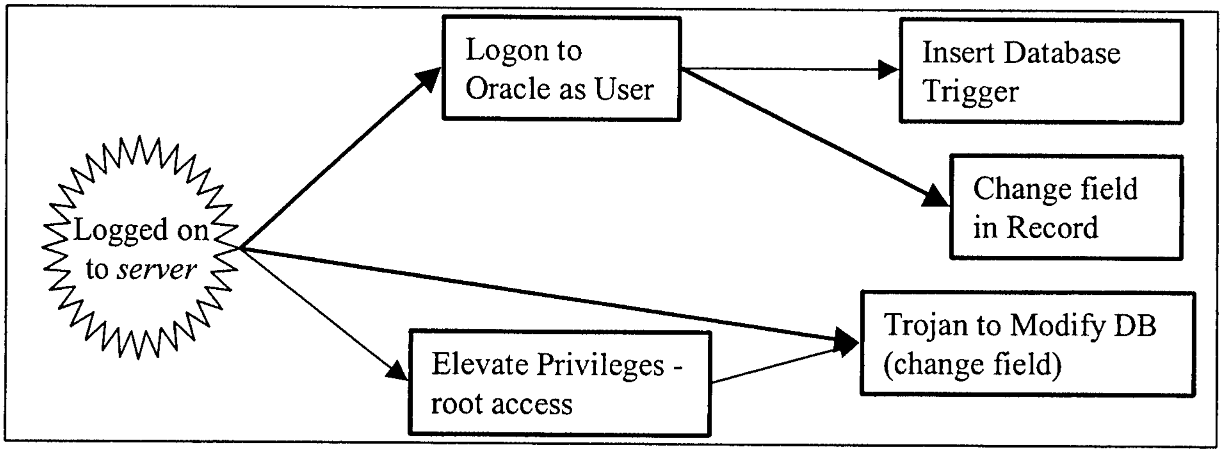

Logged on to *server*
Logon to Oracle as User
Insert Database Trigger
Change field in Record
Trojan to Modify DB (change field)
Elevate Privileges - root access

Figure 4-3. IFE 2.3 Attack Tree

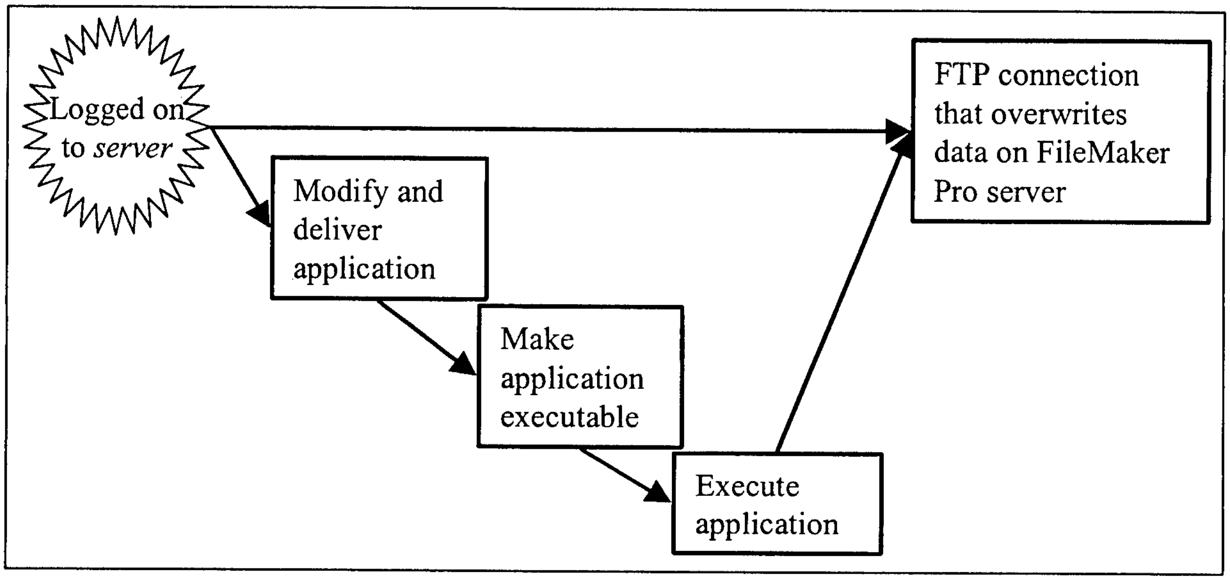

Logged on to *server*
FTP connection that overwrites data on FileMaker Pro server
Modify and deliver application
Make application executable
Execute application

Figure 4-4. IFE 2.3 Attack Tree

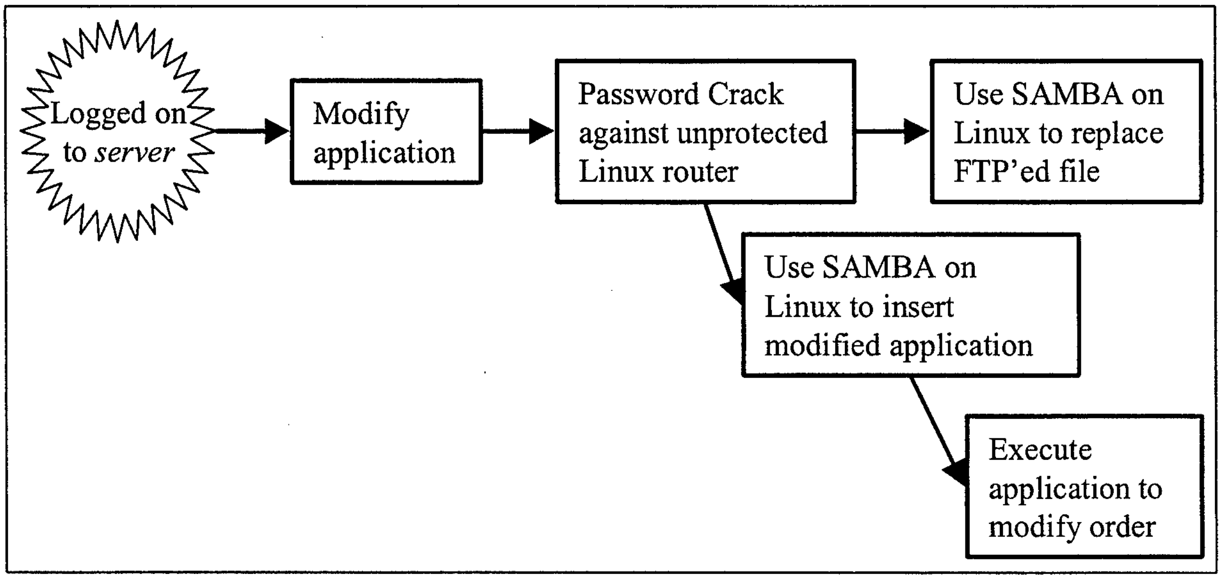

Logged on to *server*
Modify application
Password Crack against unprotected Linux router
Use SAMBA on Linux to replace FTP'ed file
Use SAMBA on Linux to insert modified application
Execute application to modify order

Figure 4-5. IFE 2.3 Attack Tree

Most of the red team attack trees started with using the sniffed password to TELNET into the Oracle/web server. A few attacks were defined that did not require this. During the IFE, the only successful attacks were those that used the sniffed password as the other attacks require a significant amount of tuning and debugging.

4.2.2.2 Detection and Response

There were no detections and hence no responses.

4.2.2.3 Issues

Although the system had a significant amount of intrusion detection capability integrated with IDIP, there were no intrusions reported. The flaw in the experiment design was the TELNET login (in the clear) to the Oracle/web server with subsequent login to Oracle. The red team simply needed to sniff the user IDs and passwords as they were sent across the *Internet Cloud*, and use these to login as a *legitimate* user and perform the modifications needed to carry out their mission. A number of small changes were made to the scenario during the IFE, but each change still permitted the red team access to a valid user account on the Oracle/web server with no detection.

The conclusion from this was that providing the adversary login access to a critical server leads to almost certain compromise unless the server is very well instrumented. This instrumentation must go beyond standard intrusion detection because red teams can work around the detection capabilities available. In this scenario, the red team did not need root access to accomplish their objectives, so the one event that a detector might catch was not required. The type of instrumentation that would have caught the red team is monitoring for everything but the limited set of operations expected from the scripted access. That is, because this was trying to model a production server, there are very limited things that are expected to occur on the server. All other actions can be considered anomalous.

4.3 IFE 2.3 Rerun

The IFE 2.3 rerun was designed to attempt to gain some insight into how we can achieve effective intrusion detection and automated response within the constraints of the IFE 2.3 configuration, applications, and scenario. The one change made was that the database account used by the consultant was not supposed to be authorized to modify the database.

Using this same configuration required development and integration of detection capabilities for detecting insiders on the Oracle/web server and the FileMaker Pro server, as well as unexpected actions visible on the network for communication to these components. The rerun was much more successful in that there were many detected events, however, the response policies were not optimal for the situation, which led to less effective responses than expected. This was a result of changes to response policy at the start of the experiment execution based on a concern that the planned response policy would greatly increase the red team's effort. A number of configuration problems also hampered the IFE 2.3 rerun, which limited the results.

In spite of the problems, the automated responses did affect the red team's behavior, and it is clear that with a simple policy file change, the red team would have had a much more difficult time reaching their flag.

The following sections describe the integration for, and execution of, the IFE 2.3 rerun.

4.3.1 Integration

Following the IFE, a small number of additional *lightweight* detectors were defined to monitor the system for any action outside the limited set expected of the operational environment. These detectors were developed to address the insider threat presented by the red team' ability to sniff a password on the Internet and use that password to gain access to the Oracle/web server as a privileged user. Most of these new detection capabilities were used in a rerun of IFE 2.3 held in December 1999. Detectors developed for the rerun include the following.

- An EMERALD extension to detect running of programs outside a list of authorized programs for a user.
- A BSM audit trail parser that detects running of programs outside a list of authorized programs for a user as well as installation of a sniffer on the local host. (This tool was not used in the rerun, as it was incompatible with EMERALD.)
- String searching in NetRadar to detect users issuing commands outside a list of authorized programs to be used across the network.
- NAI Generic Software Wrappers for Solaris controlled execution wrapper that detects running of programs outside a list of authorized programs for a user.
- ISI NT Wrappers that detects running of programs outside a list of authorized programs for a user.

- Database order detector that detects order inconsistencies.
- Oracle log reader that detects Oracle login failures.
- ifstatus that detects sniffers on a local Solaris host.
- An MD-5 hash detector that detects changes to critical files.
- A new capability built into IDIP that allows IDIP to aggregate events at an IDIP component and generate new events from the aggregate. (E.g., several attempts to use unserviced ports on a firewall or host within a short period of time could be aggregated into a port scan event.)

An additional RealSecure was also installed on the Internet Cloud to provide some visibility into red team activities on that network.

Most detectable events from the red team actions were detectable by monitoring for use of commands outside a list of allowed commands. The objective was to use multiple components to perform this monitoring to limit the red team's ability to control a single sensor. EMERALD BSM, NAI Generic Software Wrappers, and NetRadar all required some modification to support this monitoring. The plan was to use all three; however, some incompatibilities between EMERALD BSM and NAI Generic Software Wrappers made running these concurrently infeasible for the IFE. (Subsequently, the incompatibilities were isolated and fixed, so that they should now be compatible.) This forced us to perform two sets of runs: one with EMERALD BSM and one with NAI Generic Software Wrappers.

Beyond the new detection capabilities, the timing of the IFE rerun enabled us to use of some newly develop host-based response mechanisms. These include—

- Kill process or session
- Disable user account
- Disable service (e.g., Telnet)
- Reboot
- Halt

Use of these responses is controlled by a new IDIP policy file that specifies which response is appropriate for which attack. For example, when a user account is determined to be compromised (e.g., it is being used for some other exploit), the local policy file can be used to cause the user session to be killed and the user account to be disabled. If that user were remotely logged into the system, then killing the user's session also kills the connection; otherwise, it just kills the user's login session.

4.3.2 Experiment Execution

In the end, IFE 2.3 modeled a realistic environment with careless application developers, users, and system administrators. During the course of the rerun, the consultant still passed a user name and password in the clear, the problem that plagued the initial run of IFE 2.3. The new detectors compensated for that. In addition, there were problems with new services that were not supposed to be there being enabled on the firewall, resulting from careless administration. There were also some weak passwords (e.g., user name same as password) left by careless developers.

4.3.2.1 Red Team Attacks

The red team used the same attack trees for the IFE 2.3 because the basic system had not changed. The primary easy paths into the system (sniffed user password) were still present. In running the attacks, the red team exercised most of the attack tree branches used in the initial IFE 2.3. The new detection capabilities were able to see most of the red team activities. The RealSecure on the *Internet Cloud* was able to determine both that the red team had caused the Consultant Home Office workstation to crash and was then using that workstation's IP address. After the red team had logged into the Oracle/web server, the detectors were able to see the red team's use of commands that were not allowed for the user on that server.

4.3.2.2 Detection and Response

Using this new local response mechanism, we recommended that the response policy for a compromised user account (the symptom being inappropriate system use) be to disable the user account. Unfortunately, during IFE setup, it was decided that the response in this situation would only kill the process, which only kills the program identified by the detector. For command line commands, this only kills the process spawned by the shell, leaving the user session intact. In some circumstances, the kill session response was configured as the response. This kills the user's login session, which for remote users also kills the connection. This should have been used in all cases where the kill process response was specified, as a malicious child process is indicative of a malicious parent.

During the IFE, the new detectors generated a large number of attack reports in both the detector-only and detector, plus response, *runs*. The local responses had the greatest impact on the red team's behavior. By killing the process, and in some cases the session, the response slowed red team progress. The red team had to connect back to the server several times, increasing their exposure. Note that had the account been disabled and the session killed on the first offense, the red team would have required another path into the system. As with the initial IFE run, the only attacks that succeeded were ones that began with using the sniffed password to TELNET to the Oracle/web server. Thus, disabling the compromised user account would have greatly increased the red team's work factor.

One final note about the IFE 2.3 execution is that the Discovery Coordinator's cost model was not running during the IFE 2.3 rerun (a configuration over-sight). Thus, all responses were the short-term responses performed by the firewalls and routers. These were set to be very short in duration (2 minutes) for all blocking actions. With a harsher policy at the firewalls and routers, or with the Discovery Coordinator cost model running, the red team would have been further slowed.

4.4 Conclusions

IDIP response mechanisms successfully altered red team behavior. IDIP could have been more effective against the red team with a better policy. It should also be noted that the blue team was not permitted to adapt to the red team based on the attack reports, while the red team could adapt to the defenses. In a more realistic situation, the blue team would be able to adjust the policy if the existing one were to be found to be ineffective. Thus, although the experiment provided no hard data on the value of automated responses, the experiment did indicate that automated responses have the potential to be an effective defensive layer. They are certainly not a substitute for good system security design and administration. However, even in a poorly designed and administered system, the automated response system can help slow the adversary, while the centralized collection of intrusion alerts at the Discovery Coordinator provide the system operators with the information needed to take actions that are outside the purview of the automated response system (e.g., make policy changes that tighten the automated response policy).

One result is clear from the IFE 2.3 rerun: there is a need for IDIP agents to adjust their responses to situations when the Discovery Coordinator is not available. The adjustments could include lengthening the duration for timed blocking rules, or taking harsher actions (e.g., disabling TELNET at the firewall rather than disabling TELNET between two end points at the firewall). A second kind of adjustment to how IDIP operates would be to aggregate the attack data at the Discovery Coordinator to help decide when the system should globally take harsher responses. For example, when the system is under a prolonged attack, all non-essential services could be disabled. Both of these new mechanisms could range in complexity depending on the complexity of the decision algorithm. At the simple end, one could develop straightforward extensions to the existing code base that trigger off of specific events or event counts. At the complex end, one could integrate a complex reasoning tool to provide better situation assessment.

5. CIDF DEMONSTRATION

The June 1999 CIDF demonstration was used as a test of CISL maturity. Specific objectives were to-

- Test ability to create interoperable CIDF components.
- Validate utility of CISL as a language for enabling correlation
- Demonstrate some simple correlation that shows value-added created by sharing analysis results. The focus was on reducing the number of false positives and raising the number of detections through analysis of detection results across the system.

Figure 5-1 shows the configuration for the CIDF demonstration. Each detection component in the demonstration was wrapped with an IDIP wrapper and integrated via IDIP into the system. Correlation components were integrated at the Discovery Coordinator where all attack reports were collected and distributed to these correlators. The following components were integrated with IDIP to support the demonstration.

- Detectors
  - SRI EMERALD BSM Analyzer
  - ORA CORBA Anomaly Detector
  - Oregon Graduate Institute StackGuard
  - Net^2 NetRadar
  - NAI CyberCop Server
  - Internet Security Systems (ISS) RealSecure
- Correlators
  - Silicon Defense Corroborator
  - Stanford Complex Event Processing (CEP) Correlator

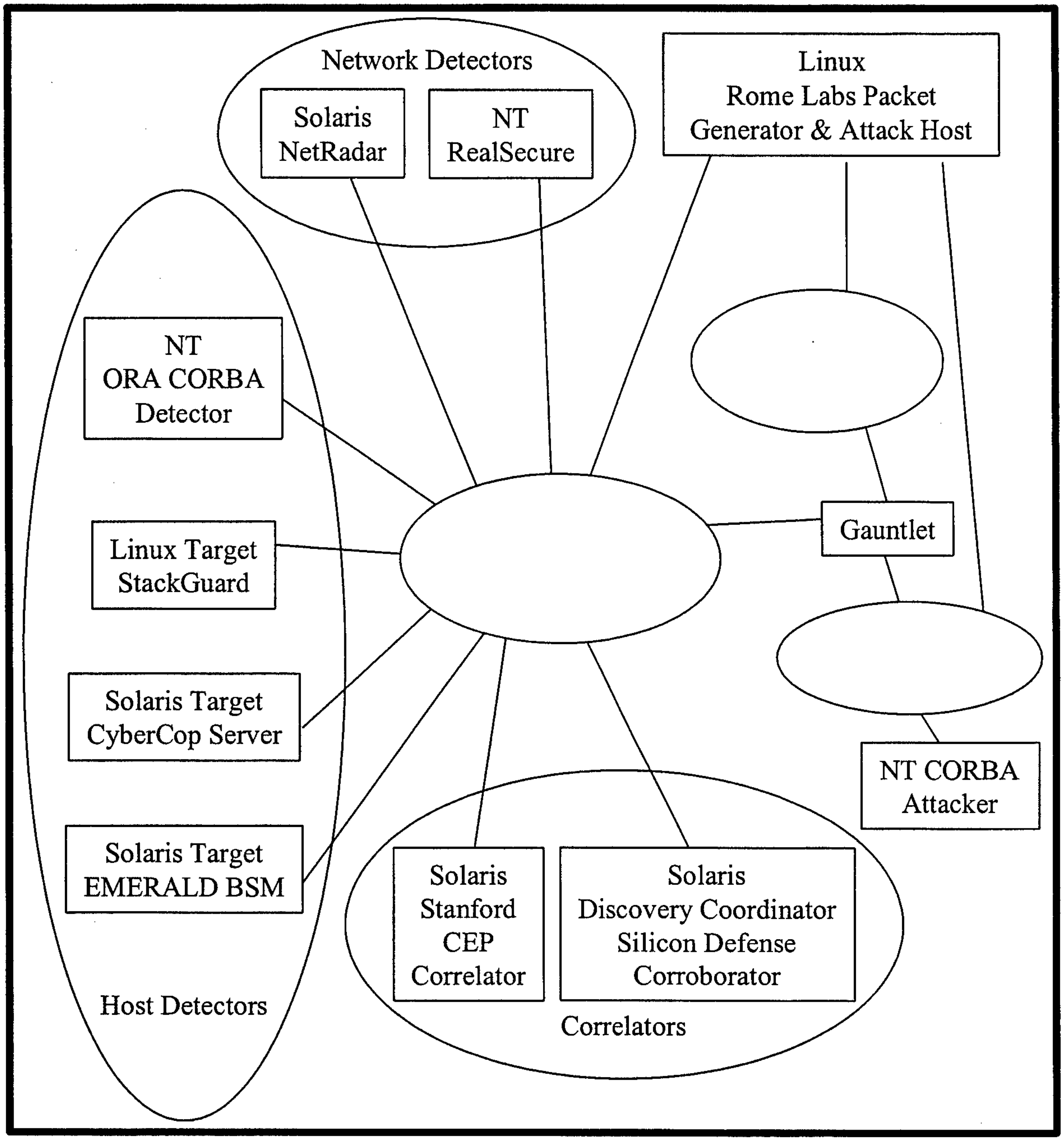

Network Detectors
Solaris NetRadar
NT RealSecure
Linux Rome Labs Packet Generator & Attack Host
NT ORA CORBA Detector
Linux Target StackGuard
Solaris Target CyberCop Server
Solaris Target EMERALD BSM
Host Detectors
Gauntlet
NT CORBA Attacker
Solaris Stanford CEP Correlator
Solaris Discovery Coordinator Silicon Defense Corroborator
Correlators

Figure 5-1. CIDF Demonstration Configuration

The demonstration successfully accomplished the objectives through successful system integration and development of algorithms by Stanford and Silicon Defense for using corroboration to reduce false positives. Using heterogeneous detectors and the IDIP infrastructure that collects attack reports at the Discovery Coordinator supported improved detection rates.

6. SUMMARY AND CONCLUSION

The focus of this project was to develop and assess mechanisms for intrusion detection and response policies. Two different types of policy mechanisms were explored: cost-based and response constraints. Cost-based mechanisms were developed for detectors, responders, and the Discovery Coordinator; each focused on a different aspect of the problem.

6.1 Lessons-Learned

The following paragraphs summarize key lessons-learned from this effort.

a. **Integration environment.** Developing an integration environment that isolates applications from changes in message layer or application layer protocol format changes greatly reduced integration costs as changes were made to IDIP. We were able to transition from the original IDIP message format to the CISL format with no application software changes.

b. **Low-cost integration strategies.** The string API developed to support integration of detection components greatly reduced the cost of integrating new detectors. The initial integration of NetRadar with this API took only 8 hours. The major effort in integrating is determining how the detector's events map into the CISL-defined events. The cost for non-IDIP developers to integrate a detector has also been very low. An Information Assurance Integration team member who was unfamiliar with IDIP integrated CyberCop Monitor, and the integration took less than a week. The generic IDIP agent and the IP audit package also reduce the cost of integrating new response components. Very little functionality must be added to produce a response component that can support tracing and blocking attacks.

c. **Loosely coupled messaging architecture.** The use of a simple message-based architecture eased the integration efforts and allowed easy evolution as we changed various parts of the protocol. Each message carries all of the state information required for a component to determine the correct response, which eliminated inter-component dependencies. This architecture also reduced the timing dependencies to those that exist at the message layer. The use of the message passing architecture has allowed easy insertion of new capabilities at the Discovery Coordinator. Because messages are multi-cast to all Discovery Coordinator clients, new functionality can be added with minimal impact on existing functions. Components can add new messages to be sent across the IDIP backplane, and because only the processes that register for the messages receive the messages, existing components need not be modified. This architecture enabled us to integrate two correlators into this infrastructure for the CIDF demonstrations with just a few hours of work.

d. **Language Flexibility.** Use of CISL as a language for describing attacks has proven useful in adding new features, as it is relatively easy to add a new feature to the attack description to support new detection and response functionality.

e. **Experiment design.** The initial IFE 2.3 failed to produce useful results for a number of reasons. The most significant lesson learned from the IFE was that for a red team experiment, it is critical to apply an assurance methodology to ensure that the attack paths available to an adversary are covered by either prevention or detection components. Failure to assess the system's assurance can leave large gaps for the red team to accomplish their mission. These gaps are paths into the system where there were no preventative measures and no detection components to compensate for the lack of prevention mechanisms. A second major lesson learned from the IFE is that there needs to be sufficient time for integration and testing prior to the experiment. Without sufficient testing, the resulting experiment can fail because functionality does not perform as needed. This IFE also pointed out the need for a "white" team in a red team exercise to help prevent the two sides from following counter-productive threads. The IFE 2.3 rerun also showed the need for using vulnerability assessment tools and managing the system configuration so as to not open holes closed during the initial system design and deployment. Another lesson is that selecting an appropriate response is critical to the effectiveness of the response system. Had the experiment used the initially proposed policy, the red team work factor would have been much higher.

6.2 Further Investigations, Research, and Development

During the course of this project, several additional response strategies and mechanisms were identified, but not developed. Implementing some of these would provide additional insight into how to effectively respond to sophisticated adversaries. These are listed below.

a. Development of a language for controlling component-specific responses. Through the investigation of host-based response mechanisms, particularly the use of NAI's Generic Software Wrappers, a number of alternative responses were identified but not implemented. Because these responses are not the same as the generic IDIP responses, the current IDIP control mechanisms provide no control over their use. Controlling their use requires a rich enough language that can be extended as new responses are identified. This also requires some general mechanisms in the Discovery Coordinator to provide users and other applications the capability to specify policy for these responses.

b. Development of the policy projection capability for the current IDIP cost models. Work towards this goal is currently being pursued on the Information Assurance Integration Contract.

c. Development of a policy projection capability for rule-based response policies such as those used in Mountain Wave's Adaptive Network Security Management project [27]. These rules are similar to the response constraints we developed, but are used to specify the desired responses for each situation instead of just limitations on responses. One issue that should be investigated is which strategy is more manageable. Our analysis indicated that the strategy for specifying policy in [27] requires the user to enter a large number of cases describing each situation or class of situations. The policy mechanisms defined under our project require entry of more abstract information, because many situations are covered by the cost parameters. The risk is that the user may not be able to characterize the values used in these models as easily as the more concrete policies specified for the Adaptive Network Security Management "Recon" system.

d. Development of alternate response strategies. Several response strategies have been identified in addition to the ones represented by the current implementation. Developing some of these would provide additional insight into the effectiveness of automated response. For example, the one strategy implemented for Discovery Coordinator loss is a strategy that escalates the response for repeated attacks. This should migrate the response towards the attacker as components nearest the attacker will take more severe responses: their blocking rules prevent other nodes from seeing the repeated attacks. An alternate strategy would be to increase communication between agents to determine whether a node closer to the attack source is already blocking, in which case local blocking would be removed. This can be done with a small number of additional messages (and was actually an unused feature of the initial IDIP specification). Another strategy would be to replace the Discovery Coordinator with more sophisticated agents that use agent technology to support distributed complex reasoning.

e. Developing a more dynamic version response system. The current IDIP implementation has a number of control mechanisms for configuring statically deployed IDIP agents. The current implementation allows the Discovery Coordinator to push this configuration information to the IDIP agents when either policy or configuration changes are required. However, there is no capability to push out new IDIP agents or new functionality. This type of capability would allow systems to modify their behavior in real-time as new response strategies and mechanisms are developed in response to new adversary behaviors. Agent technology is one method of achieving this type of functionality.

f. Related to the previous item, NAI Labs, with Boeing, has recently begun an investigation of how to use active network technology to provide a dynamic response capability. This new networking paradigm requires different strategies for response because packets are executed at each node instead of simply being routed through the network. This presents new challenges for intrusion detection and response, but active networks also provide a new capability of having response agents that traverse the network themselves.

g. Addition of mechanisms to support inter-community cooperation. Boeing, with NAI Labs, Silicon Defense, and U. C. Davis, is developing this capability under another recently begun research effort. The issues to be investigated include (1) determining what information should be shared between domains with different relationships (peer or hierarchical) to support both correlation and response; (2) developing mechanisms to support this controlled sharing; (3) determining algorithms to correlate information across domain boundaries; (4) developing strategies for managing trust in a multi-domain environment.

6.3 Conclusion

We have developed an initial set of intrusion detection and response policy mechanisms that enables system administrators to constrain the response system so that responses do no more damage to system functionality than the attacks against which they are responding. We have also developed a policy management architecture that can support system administrators providing high-level policy directives, with a policy projection mechanism that translates these abstract policy commands into more detailed policy configuration information to be used by IDIP agents and the Discovery Coordinator. The policy mechanisms use a value system used to evaluate the cost of attacks and responses. We have also made major contributions to CIDF through providing the CIDF message layer and providing input to CISL to support response features.

The mechanisms developed support the original IDIP requirements, as well as providing better control over IDIP response.

a. **Real-time response.** The IDIP response components use simple cost-benefit models to determine an appropriate short-lived response that attempts to eliminate the attack quickly. The algorithm can be made as efficient as standard routing algorithms. The short-term response provides time for more sophisticated algorithms to determine better response.

b. **Multiple administrative domains.** The mechanisms defined allow each administrative domain to operate autonomously with minimal knowledge of remote domains. We have identified some of the issues involved in supporting sanitization of requests and adjusting responses from other domains to allow minimally cooperating domains to respond to intrusions that cross domain boundaries.

c. **Minimal system performance impact.** After initialization, IDIP consumes minimal network bandwidth when there is no attack activity. Only infrequent keep-alive messages are used to maintain the neighborhood state. During attacks, use of multicast operation and relatively compact messages minimizes affects on network resources. IDIP-invoked filtering of network traffic, which can cause boundary controller performance degradation, is designed to be relatively short-lived, so that parts of the network far removed from the attack source have only short-term affects once the Discovery Coordinator has implemented the optimal system response. The IDIP mechanisms also only place responses along the attack path (or for Discovery Coordinator-generated responses, along alternate paths available to the attacker).

d. **Operating while the system is under attack.** The use of a lightweight, secure, reliable UDP for communication reduces the effects of attacks on IDIP.

e. **Autonomous response.** The mechanisms defined allow each IDIP node and each Discovery Coordinator to independently determine their responses based on the IDIP messages and policy parameters.

f. **Scalability.** IDIP has been used with moderate sized neighborhoods, and in IFE 2.3

    1. Scalability from small to very large systems. The use of IDIP “neighborhoods” (see Section 2) limited the knowledge required of each IDIP component enabling easy growth of the IDIP system. IDIP components only have to know about other IDIP components that are nearby, plus their discovery coordinator. This reduces the management required for each component.

    2. Robustness. IDIP was able to continue intrusion response operation even during attacks that flooded the network or slowed down IDIP components. Additional work, however, could further improve this robustness.

3. End-user transparency. No application changes were required to allow IDIP operate. IDIP uses minimal network resources. When the system is under attack, IDIP network utilization increases to support tracing and centralized reporting, but the number and size of messages is still relatively small.
4. Simplicity. The IDIP application layer was particularly simple to implement and integrate with the IDIP component prototypes. The use of a simple User Datagram Protocol (UDP) based protocol enabled quick development of the IDIP message layer. The primary complexity in IDIP is the key management functions required to provide IDIP self-protection.
5. Protection against spoofing. IDIP cryptographic services provide protection against the spoofing of IDIP components through authentication and integrity mechanisms for IDIP messages. We use a standard keyed hash algorithm to support authentication within each IDIP neighborhood.
6. Compatibility with multiple encryption technologies. Although the initial IDIP implementation uses Fortezza hardware, IDIP's cryptographic mechanisms are algorithm independent, enabling integration of additional cryptographic algorithms to support varying user requirements. The current implementation uses the same mechanisms to support a software-based set of cryptographic algorithms shown in Figure 6-1.

Cryptographic Service	OpenSSL Algorithm
Data Privacy	DES
Data Integrity	SHA-1
Digital Signature	DSA
Key Exchange	El Gamal (Modification of OpenSSL's Diffie-Hellman)

Figure 6-1. Current Cryptographic Algorithms

We found fewer alternative policy mechanisms than originally anticipated. Beyond the currently implemented mechanisms, we identified another policy alternative to cost models where the responses are prescribed for each potential situation. This is the approach taken in [27]. Our assessment was that this alternative policy approach would require many more policy statements to be entered by system administrators and more complex algorithms at the IDIP agents to implement the policy.

The IFE 2.3 rerun showed that automated response technology has promise in providing protection against adversary attack provided the combination of prevention and detection mechanisms do not provide the adversary with easy undetectable access to critical system resources. New IDIP-related mechanisms would be useful for handling aggregated attack data and to enable the IDIP agents to be more effective when the Discovery Coordinator is not active.

7. REFERENCES

[1] Network Associates, Gauntlet Firewall, http://www.nai.com/asp_set/products/tns/intro.asp.

[2] Secure Computing Corporation, Sidewinder, http://www.securecomputing.com/.

[3] Linux, http://www.linux.org/.

[4] TCP Wrappers http://cs-www.ncsl.nist.gov/tools/tools.htm

[5] IP Filter, http://coombs.anu.edu.au/~avalon/ip-filter.html.

[6] T. Fraser, L. Badger, M. Feldman, "Generic Software Wrappers "Hardening COTS Software with Generic Software Wrappers", Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE, Oakland, California, May 1999.

[7] J. Epstein, "Architecture and Concepts of the ARGuE Guard", to be published in Proceedings of the 15th Annual Computer Security Applications Conference, Phoenix AZ, December 1999.

[8] G. Lamperillo, "Architecture and Concepts of the MPOG", NAI Labs Reference Number #0768, June 1999.

[9] Net Squared, Network Radar, http://www.NetSQ.com/Radar/.

[10] Ulf Lindqvist and Phillip A. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)", Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE, Oakland CA, May 1999.

[11] C. Cowan, C. Pu, D. Maier, H. Hinton, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks", Proceedings of the 7th USENIX Security Conference, San Antonio TX, January 1998.

[12] Odyssey Research Associates, Inc., "Computational Immunology for Distributed Large Scale Systems", http://www.oracorp.com/Projects/Current/CompImm.htm.

[13] Network Associates, CyberCop Intrusion Protection, http://www.nai.com/asp_set/products/tns/intro.asp.

[14] Internet Security Systems, RealSecure, http://solutions.iss.net/products/rsecure/rs.php.

[15] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle, "GrIDS -- A Graph-Based Intrusion Detection System for Large Networks", Proceedings of the 19th National Information Systems Security Conference, October 1996.

[16] L. Perrochon, W. Mann, S. Kasriel, and D. C. Luckham, "Event Mining with Event Processing Networks", Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining. Beijing, China, April 1999.

[17] Rich Feiertag, Cliff Kahn, Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Brian Tung, "A Common Intrusion Specification Language", http://www.gidos.org/, June 1999

[18] Clifford Kahn, Don Bolinger, Dan Schnackenberg, "Communication in the Common Intrusion Detection Framework, v 0.7", http://www.gidos.org/drafts/communication.txt, June 1998.

[19] The Boeing Company. *Neighborhood Key Information Distribution (NKID) Protocol (Draft)*, Boeing Document Number D658-10818-1, February 1998.

[20] Trusted Information Systems, Inc. *Intruder Detection Isolation Protocol (IDIP) Authentication Header (AH)*, TIS Report Number 0699D, November 1997.

[21] Trusted Information Systems, Inc., *IDIP AH with Hashed Message Authentication Codes (HMAC)-SHA-1,* TIS Report Number 0700D, November 1997.

[22] Trusted Information Systems, Inc., *Intruder Detection Isolation Protocol (IDIP) Encapsulating Security Payload (ESP)*, TIS Report Number 0698D, November 1997.

[23] Trusted Information Systems, Inc., *IDIP ESP with SKIPJACK Cipher Block Chaining (CBC),* TIS Report Number 0701D, November 1997.

[24] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", Network Working Group, Request for Comments 2401, November 1998

[25] OpenSSL Project, http://www.openssl.org/.

[26] LBNL's Network Research Group, "libpcap, the Packet Capture library", http://ee.lbl.gov/.

[27] Mountain Wave, Inc., "Adaptive Network Security Management", http://www.mountainwave.com/darpa-report/.

A. APPENDIX A MESSAGE LAYER API SPECIFICATION

This appendix describes each exported call provided by the CIDF Message Layer API. This is for the benefit of both application programmers who will use these calls as well as implementers who want to provide this API to others.

Unless otherwise noted, calls return 0 on success.

A.1 Definitions

The following definitions specify maximum values for message layer functions.

```
#define MAXCIDFGROUPS      10      /* Maximum number of multicast groups */

#define MAXCIDFMSGLEN      65400   /* Maximum application message size */

#define MAXCIDFCLASSES     16      /* Maximum number of classes to which an application
                                      can subscribe */
```

The following global variable is used to store human readable error messages from the Message Layer. When the Message Layer returns an error code, it also writes to this string.

```
char CIDFerrstr[80];
```

The CIDF Message Layer requires a configuration file that holds multicast group information. This information will eventually be replaced through use of a CIDF directory service.

The configuration file format is as follows:

```
#
# This is the configuration file. It must be global across all
# CIDF components. (lines starting with "#" are comments)
#
# Note: Hx is either a hostname or a IPv4 unicast address
#
group all: \
        multicast = 224.1.2.3: \
        member = H1, \
        member = H2, \
        member = ..., \
        member = HN
group response: \
        multicast = 224.1.2.4: \
        member = H2, \
        member = H4, \
        member = H6
group analysis: \
        multicast = 224.1.2.5: \
        member = H1, \
        member = H3, \
        member = H6
neighborhood external_lan : \
        multicast = 224.1.2.6: \
        member = Ha, \
        member = Hb, \
        member = Hc
neighborhood internal_lan : \
        multicast = 224.1.2.7: \
        member = Hd, \
        member = He, \
        member = Hf
```

A.2 Start-Up Calls

A.2.1 CIDFMLinit

Syntax:

```
int CIDFMLinit( char* filename )
```

This call reads in the specified CIDF configuration containing the multicast group information and initializes the Message Layer. If the filename is NULL, this function reads the default file "./cidf-ml.ini" is used. Applications must call CIDFMLinit prior to calling other CIDF Message Layer functions.

Return values:

CIDFML_PARAM_ERROR is returned under the following conditions:

filename is NULL and "./cidf-ml.ini" does not exist

CIDFML_RESOURCE_ERROR is returned on insufficient resources to

initialize or socket call failure

A.2.2 CIDFMLbind

Syntax:

int CIDFMLbind(char * groups[], unsigned int groupcount)

Subscribe to a CIDF group or list of groups (specified in the configuration file) and return a handle to be used in subsequent calls referencing this list of groups.

Return values:

handle (if return is >= 0)

CIDFML_PARAM_ERROR is returned under the following conditions:

groupcount > MAXCIDFGROUPS

groups is NULL

CIDFML_RESOURCE_ERROR is returned on a generic resource error

A.2.3 CIDFMLrestrict

Syntax:

int CIDFMLrestrict(int handle, ushort * classid, unsigned int classcount)

Restrict delivery of messages to only those messages that match one of the class IDs specified in classid. These class ID are defined in CISL Section 6.3.

CIDFMLrestrict maybe applied to the handle at any time, however the restrictions do not apply to messages that are in the process of being delivered. That is, there is no guarantee that only these classes will be delivered until currently queued messages have been delivered.

Only the most recent CIDFMLrestrict is applied to the handle.

Return values:

CIDFML_PARAM_ERROR is returned under the following conditions:

handle is not valid

classcount > MAXCIDFCLASSES

classid is NULL

CIDFML_RESOURCE_ERROR is returned on a generic resource error

A.3 Communication Calls

A.3.1 CIDFMLrecvfrom

Syntax:

```
int CIDFMLrecvfrom( int handle, void* data, unsigned int max_data,
                    unsigned int timeout, char * group)
```

Receive a CIDF message. Using a handle returned from CIDFMLbind, wait up to timeout seconds for a message to be copied (up to max_data bytes) into data). If group is non-null, the ASCII group name associated with the received message is copied into group. A timeout of 0xFFFFFFFF causes the application to block until the next message arrives.

Only messages from the groups associated with the handle are delivered, and only if they match the constraints specified in CIDFMLrestrict. Messages queued for delivery prior to a CIDFMLrestrict call may not match these constraints.

This function strips off the Message Layer header and applies the cryptographic mechanisms prior to message delivery.

Return values:

actual bytes written to data (if return is >= 0)

CIDFML_PARAM_ERROR is returned under the following conditions:

handle is not valid

data is NULL

max_data > MAXCIDFMSGLEN

CIDFML_RESOURCE_ERROR is returned on a generic resource error

A.3.2 CIDFMLsendto

Syntax:

```
int CIDFMLsendto( char * group[], unsigned int groupcount, void* data,
                  unsigned int length)
```

Send a CIDF message comprised of length number of octets from data to the group (or group list) specified. The message is transmitted to each host that is a member of the group (or group list).

This function adds the Message Layer header and applies the cryptographic mechanisms prior to message transmission.

Return values:

actual bytes copied from data (if return is >= 0)

CIDFML_PARAM_ERROR is returned under the following conditions:

group is NULL

groupcount > MAXCIDFGROUPS

data is NULL

length > MAXCIDFMSGLEN

CIDFML_RESOURCE_ERROR is returned on a generic resource error

A.4 Termination Calls

A.4.1 CIDFMLclose

Syntax:

`int CIDFMLclose( int handle )`

Unsubscribe to the CIDF group or list of groups using the handle returned from CIDFMLbind. The application may still receive messages from other groups using other valid handles.

Return values:

CIDFML_PARAM_ERROR is returned under the following conditions:

handle is not valid

CIDFML_RESOURCE_ERROR is returned on a generic resource error

A.4.2 CIDFMLexit

Syntax:

`void CIDFMLexit(void)`

Release CIDF Message Layer resources. CIDF Message Layer calls from this process will fail following this functions.

Return values:

None

A.5 Error Codes

The following error code values are to be used by API callers and implementers.

```
#define CIDFML_SUCCESS              0
#define CIDFML_PARAM_ERROR         -1
#define CIDFML_RESOURCE_ERROR      -2
```

©U.S. GOVERNMENT PRINTING OFFICE: 2002-710-038-10239

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.